Universidad Central "Marta Abreu" de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Agentes Inteligentes para la identificación de sistemas en procesos biotecnológicos de tratamiento de residuales

Autor: Billy Antonio Gácita González

Tutor: MSc. Ing. Iván Iglesias Navarro

Dr. Ing. Boris Luis Martínez Jiménez

Dr. Ing. Francisco Herrera Fernández

Santa Clara

2012

"Año 54 de la Revolución"

Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Agentes Inteligentes para la identificación de sistemas en procesos biotecnológicos de tratamiento de residuales

Autor: Billy Antonio Gácita González

e-mail: bgacita@uclv.edu.cu

Tutor: MSc. Ing. Iván Iglesias Navarro

e-mail: iglesias@uclv.edu.cu

Dr. Ing. Boris Luis Martínez Jiménez

e-mail: boris@uclv.edu.cu

Dr. Ing. Francisco Herrera Fernández

e-mail: herrera@uclv.edu.cu

Santa Clara

2012

“Año 54 de la Revolución”

Hago constar que el presente trabajo de diploma fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería en Automática, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Autor

Firma del Jefe de Departamento donde se defiende el trabajo

_____

Firma del Responsable de Información Científico-Técnica

PENSAMIENTO

Bienaventurado el hombre que halla la sabiduría y obtiene inteligencia, porque su ganancia es mayor que la de la plata, sus beneficios más que los del oro refinado.

Proverbios 3: 13, 14

DEDICATORIA

A todos los que sin importar el sacrificio me ayudaron y han estado conmigo en momentos buenos y difíciles.

A los que estuvieron comprometidos con el desarrollo del trabajo.

A mi bebé que pronto nacerá.

AGRADECIMIENTOS

Sirvan estas insuficientes líneas para expresar un sincero agradecimiento a todos aquellos que han hecho posible directa o indirectamente la realización de este trabajo y mi formación como ingeniero.

Doy gracias a DIOS por permitirme llegar hasta aquí y ayudarme a conocer tantas personas que me han apoyado y enseñado de múltiples maneras.

Agradezco a mi amada esposa Lisania y a toda nuestra familia por su ayuda incondicional en todos los sentidos especialmente a mi madre Maritza y Tony, mis abuelos Zeyda y Antonio, mi padre Mario Antonio (Tonito) y Lily, mis hermanos Seydi y Alejandro, a papi y a mami, mis suegros, mis demás hermanos (personas que tienen ese significado para mí) y amigos que se identifican con el arduo trabajo que ha representado este proyecto y se preocupan por mí, a todos, muchas gracias.

A mi tutor Iván Iglesias Navarro por apartar en muchas ocasiones un lugar de su apretada agenda para escuchar mis dudas y planteamientos, además de brindarme sus criterios y sus exhaustivas revisiones que sin lugar a dudas tuvieron un gran peso en la confección del trabajo.

A mi tutor Boris Luis Martínez Jiménez, que nos regaló parte de su tiempo y aportó talentosas ideas y experiencias que representaron una gran ayuda para el avance del proyecto.

Un reconocimiento especial a todos mis compañeros de grupo, muchos de ellos me aclararon dudas durante la carrera, sin embargo hay dos nombres que estoy obligado a mencionar, más que compañeros son mis amigos, durante ocho años hemos estado juntos, ellos son Emilio y Denis. Sirva este medio también para agradecer a muchachos de otras carreras que me ayudaron muchísimo, más allá del sacrificio, alguien que sobresale es Andy.

A varios profesores como José Omar y Félix que en el transcurso de la carrera han estado dispuestos a brindarme su ayuda cuando lo he necesitado, han servido como ejemplo en varias áreas de mi vida.

Al concluir esta sección solo puedo decir:

A todos, muchas gracias…

RESUMEN

La presente investigación surge ante el poco avance en la identificación de sistemas con agentes inteligentes. Actualmente estos se levantan como el nuevo paradigma de la Inteligencia Artificial, evidenciándose en el gran número de aplicaciones en que están presentes de forma creciente. Este trabajo presenta como objetivo proponer una estructura organizacional de Sistemas Multiagentes (SMA) para la identificación de sistemas en los procesos biotecnológicos de una planta de tratamiento de residuales para ello se apoya en una exhaustiva revisión bibliográfica que estableció varios criterios y conceptos importantes para el desarrollo de la tesis. Finalmente después de estudiar varias metodologías se alcanza la propuesta del SMA anterior siguiendo la metodología INGENIAS, describiendo su organización estructural y desarrollando los modelos que describen las principales características del sistema. Este proyecto aporta un gran valor teórico para la asignatura Inteligencia Artificial (IA) y establece una sólida base para la implementación del SMA al definir la estructura y abarcar las características que mejor describen el sistema.

TABLA DE CONTENIDOS

INTRODUCCIÓN

La inteligencia artificial es considerada una de las áreas de la ciencia que más interés despierta para desarrollar novedosas investigaciones, esta disciplina está presente donde se necesita modelar el comportamiento humano para resolver problemas con cierto grado de dificultad necesitándose características asociadas con la inteligencia humana. Actualmente la incursión de los agentes inteligentes en disímiles aplicaciones de la IA es un gran paso de avance, hasta el punto de convertirse en un nuevo paradigma (Julián and Botti 2000), estos son capaces de realizar acciones autónomas y poseen una serie de propiedades que los diferencian de los programas convencionales. General Magic en 1990 fue la primera compañía en invertir en tecnología de agentes para construir un sistema basado en agentes para dispositivos portátiles (Mancilla 2008). Dentro del campo industrial existen varios sistemas de control de procesos y manufactura que utilizan agentes para llevar a cabo sus funciones, como ejemplo se puede citar ARCHON (Architecture for Cooperative Heterogeneous Online Systems) (García and Ossowski 2003). Esta es una plataforma de software para la construcción de sistemas multiagentes. A pesar de estos avances, no se reflejan en la bibliografía trabajos anteriores de agentes inteligentes usados para la identificación de estas plantas de tratamientos de residuales para modelo predictivo. En nuestro país y en nuestra universidad particularmente se han realizado trabajos dentro de la Inteligencia Artificial pero no específicamente en la identificación de estos procesos de tratamientos de residuales, es por ello que si se logra en esta rama, una aplicación de SMA, constituiría un paso de avance en la Automática de nuestro país y de nuestro centro específicamente, además se podría usar la estructura de organización y otras características adyacentes como base de estudio en algunas asignaturas de la carrera.

La **situación problémica** del presente trabajo está dada por el poco avance en la identificación con el uso de agentes inteligentes. En cuanto a lo que corresponde por parte

de la automática, la mayor necesidad está, en poseer modelos matemáticos lo más precisos y cercanos a la realidad que reflejen la descripción del comportamiento de las principales variables que intervienen en estos procesos, para obtener el logro de los mejores resultados empleando el desarrollo de SMA.

Se podría plantear lo siguiente ¿Cómo obtener una estructura de SMA para la identificación de sistemas de procesos de tratamiento biológico? Actualmente no se cuenta con una estructura organizacional para llevar a cabo la identificación de estos procesos; esto depende en gran medida de las características operacionales del proceso objeto de estudio y de las condiciones climáticas en que este se desarrolle, dando paso a las siguientes interrogantes:

- ¿Podrá emplearse un SMA en la identificación de sistemas?
- ¿Existe alguna metodología capaz de permitir el desarrollo de un SMA para la identificación de sistemas en los procesos de tratamiento de residuales?

Para dar cumplimiento a estas interrogantes la presente investigación se propone el siguiente **Objetivo General:** proponer una estructura de sistema multiagente, para la identificación de sistemas en los procesos biotecnológicos, de una planta de tratamiento de residuales. Además se toman en consideración los siguientes **objetivos específicos:**

1. Identificar en la literatura las metodologías existentes para el diseño de Sistemas Multiagentes.
2. Proponer una estructura de SMA para llevar a cabo el proceso de identificación de sistemas en procesos biotecnológicos.

La investigación está enfocada a la industria petrolera y debe aplicarse específicamente en la refinería Camilo Cienfuegos en la provincia de Cienfuegos. Esta gran refinería cubana aún cuando realiza un trabajo eficiente presenta la desventaja del vertimiento de grandes cantidades de residuales, desperdiciando así algunas sustancias como el agua, además del impacto negativo sobre el medio ambiente que rodea la industria.

Aportes: La presente investigación se convertirá en una referencia dado que la aplicación de SMA en la identificación de sistemas no ha sido frecuente. A partir del estudio de las metodologías se podrá conocer cuáles de ellas son las candidatas más fuertes para el

desarrollo de este tipo de sistemas al cumplir con una serie de requerimientos que este impone. Se establecerán bases sólidas para una futura implementación del SMA al diseñar una estructura de organización para el sistema y definir las características principales que lo describen.

Los modelos obtenidos o identificados a través del uso de agentes inteligentes, tienen la posibilidad de reutilización dada sus características de aprendizaje, sobre todo en este tipo de industria, evidenciándose así su gran implicación práctica. Con la aplicación de SMA, el informe cobra un gran valor teórico porque sería un referente para los estudiantes, mucho más ahora que en el plan D se incorpora una asignatura optativa "Inteligencia Artificial".

La **novedad científica** de este trabajo está en lograr una estructura de organización para un SMA que a partir de su desarrollo e implementación tenga la capacidad de obtener el mejor modelo matemático analizando no solamente criterios como el *FIT*, la varianza u otro que un programa convencional pueda manejar sino que tenga en cuenta además, elementos que un ser humano tuviera presente como por ejemplo: la relación existente entre exactitud requerida y grado de dificultad para obtener un modelo a partir del número de coeficientes.

Resultados esperados: Fundamentalmente a partir de una metodología la propuesta de una estructura de SMA para la identificación de sistemas en los procesos biotecnológicos de una planta de tratamiento de residuales, además de reflejar de alguna forma las principales características que describen el sistema que a la postre ayudarán a su implementación.

Organización del informe

El trabajo de diploma, posterior a esta introducción, incluye principalmente tres capítulos, las conclusiones y recomendaciones.

En el capítulo 1 se definirán varios conceptos, además de analizar variados criterios sobre inteligencia artificial, agentes inteligentes e identificación de sistemas.

En el siguiente capítulo se describen y analizan una serie de metodologías que están orientadas o incorporan el trabajo con SMA. Cada una de ellas se caracteriza atendiendo a su orientación, si poseen o no alguna herramienta de desarrollo, sus puntos débiles y

principales ventajas. Al finalizar dicha sección se justificará la selección de una de ellas para la propuesta de SMA.

En el capítulo 3 la metodología seleccionada, INGENIAS, sirve de guía para la propuesta de un SMA para la identificación de sistemas en procesos biotecnológicos de tratamientos de residuales. Haciendo uso de notaciones de INGENIAS se describen los modelos que expresan las características fundamentales del SMA.

Posteriormente puede apreciarse las conclusiones y recomendaciones a partir del análisis de varios argumentos y del cumplimiento de los objetivos trazados al inicio de esta investigación.

CAPÍTULO 1. LOS AGENTES INTELIGENTES PARA LA IDENTIFICACIÓN DE SISTEMAS.

La Inteligencia Artificial (**IA)**, se está considerando como una de las disciplinas más nuevas, siendo curiosamente considerada a la vez como una gran ciencia desconocida, pocas personas tienen claro qué es la IA, sin embargo es considerada por una gran mayoría de científicos como la disciplina donde han pensado alguna vez en trabajar. Se ha considerado por muchos que el siguiente paso en la inteligencia artificial son los agentes inteligentes (Julián and Botti 2000), estos son capaces de actuar tal y como lo hacen las personas, es decir pueden modelar el comportamiento humano, incluso son capaces de adelantarse y realizar tareas que nosotros podríamos realizar solo si dispusiéramos de más tiempo. Los agentes aún cuando están presente en muchas esferas abarcando gran cantidad de aplicaciones no han tomado auge en la identificación de procesos para el tratamiento de residuales, sin embargo, analizando distintos criterios sobre los agentes inteligentes y una serie de características que los definen se ha estimado que implementar una aplicación en dicha rama sería un paso de avance en la identificación de sistemas y para la automática de forma general. Para desarrollar esta idea es necesario partir de la base de una serie de criterios y conceptos que permitirán enfocarse en la dirección deseada para el desarrollo de la investigación, es por ello que el presente capítulo se ocupa de definir conceptos y abordar criterios relacionados con los temas de inteligencia artificial, agentes, agentes inteligentes, sistemas multiagentes, además de la identificación de sistemas, todo ello a partir de una minuciosa revisión bibliográfica que además ha permitido conocer el auge de los agentes inteligentes para el desarrollo de numerosos proyectos investigativos elaborados por importantes universidades en el mundo y prestigiosas personalidades en el mundo de la inteligencia artificial.

1.1. Inteligencia Artificial.

Existen gran variedad de definiciones respecto a lo que es la IA, algunas más válidas que otras, según Larousse inteligencia (Del lat. intelligentia.)1 Facultad para comprender o entender las cosas. 2 Cualidad de inteligible: su último libro es de difícil inteligencia. 3 Habilidad o acierto: duda de su inteligencia para ocupar este cargo de tanta responsabilidad. 4 Acuerdo o entente entre dos o más personas: entre ellos no hay una buena inteligencia.

De igual forma Larousse define artificial como: 1- Que ha sido hecho por el hombre: prefiero las flores artificiales, no me dan alergia. 2- Se refiere a la persona, acción, sentimiento que muestra artificio o poca naturalidad: me molesta su artificial sonrisa.

A partir de los conceptos citados Larousse define inteligencia artificial como: Capacidad de un ordenador o de un programa de simular comportamientos humanos, tales como el reconocimiento de la situación en la que se encuentra al operar y la mejora de las decisiones por lo aprendido.

La Inteligencia Artificial (IA), definida como el estudio de como programar computadoras que posean la facultad de hacer aquello que la mente humana puede realizar (Barber, Botti et al. 2002)

Se puede definir a la Inteligencia Artificial, nombrada por John Macarte, como el campo de las ciencias computacionales que trata de mejorar el desempeño de las computadoras al dotarlas de características asociadas con la inteligencia humana, como la capacidad de entender el lenguaje natural, o de razonar bajo condiciones de incertidumbre para tomar las mejores decisiones (Mancilla 2008).

La IA es un intento de reproducir (modelar) la manera en que las personas identifican, estructuran y resuelven problemas difíciles (Pople 1985).

Se podría intentar resumir las definiciones anteriores comentando que la IA es una disciplina científico- técnica que persigue el desarrollo de sistemas que piensen y actúen racionalmente, intentando modelar el comportamiento humano.

Ella toma un sentido científico viable, como disciplina informática moderna, durante la segunda mitad del siglo XX y es el resultado directo de la confluencia de diversas corrientes intelectuales (Barber, Botti et al. 2002) (Teoría de la Computación, Cibernética, Teoría de la Información, Procesamiento Simbólico, Psicología) desarrolladas sobre las

bases formales de la Lógica y la Matemática Discreta, e impulsadas por el desarrollo de los ordenadores digitales. Particularmente, la IA requiere un gran esfuerzo para entender la complejidad de la experiencia humana en términos de procesamiento de información y no solamente trata sobre cómo representar y usar lógicamente una compleja e incompleta información, sino que además se ocupa de áreas como la visión (ver), la robótica (moverse), lenguaje natural o habla(comunicarse), el aprendizaje, etc.

En los últimos años la IA ha ido evolucionando, quizás con mayor celeridad que otras disciplinas, motivada probablemente por su propia inmadurez. Todo esto ha llevado a que la IA actualmente abarque una gran cantidad de áreas, desde algunas muy generales como razonamiento, búsqueda, etc. a otras más específicas como los sistemas expertos y sistemas de diagnóstico, por citar algunos. Se podría indicar, sin lugar a dudas, que la IA puede ser aplicada hoy en día a infinidad de disciplinas científicas y es que la IA es susceptible de aparecer allí donde se requiera el intelecto humano. En este aspecto cada avance de IA en un campo diferente necesita de una metodología de aplicación propia del investigador que lo intenta pues a diferencia de otras disciplinas, no existen criterios consensuados que unifiquen el proceso de aproximación de la IA a problemas reales. La IA ha estado presente en diversas esferas que van desde la implementación de juegos en computadora, demostraciones de teoremas, pasando por el surgimiento de nuevas técnicas como Robótica; Sistemas Expertos; Redes Neuronales; Manipulación Inteligente de Base de Datos; Web Inteligentes; Programación Automática; Visión Computarizada hasta llegar a los Agentes Inteligentes (Blanco 2002).

Actualmente en la IA ha surgido un nuevo paradigma conocido como «paradigma de agentes» (Julián and Botti 2000) (Choque 2009), el cual ha tomado un gran auge. El novedoso paradigma plantea el desarrollo de entidades que puedan actuar de forma autónoma y razonada.

1.2. Agentes Inteligentes.

Existe una gran variedad de definiciones para el término agente, el Gran Diccionario de la Lengua Española Larousse lo define como 1 Que obra o puede obrar. 2 Persona o cosa que produce un efecto. 3 Persona que gestiona ciertos asuntos en nombre y beneficio de otra. 4 Persona que tiene a su cargo una agencia que presta determinados servicios. 5 Empleado

municipal o gubernativo que se encarga de velar por la seguridad pública o por el cumplimiento de las leyes.

Varios investigadores y grupos de investigación han definido el término de agente desde diferentes puntos de vista, esto ha influido a que en la actualidad existan diferentes definiciones de lo que es un agente, la dificultad se debe especialmente a que los agentes se pueden emplear en muchos dominios de aplicación dentro de los cuales se encuentra la biotecnología, la electrónica y la informática, llevando consigo a que cada dominio asocie varios y diferentes atributos a la definición de agente.

Un agente es un sistema que está en algún ambiente y que es capaz de tomar acciones autónomas de acuerdo al estado del ambiente para cumplir sus objetivos de diseño (Choque 2009).

Otra definición es la que da (Russell and Norvig 1996) en la que dice que: Un *Agente*, de manera general, es todo aquello que puede considerarse que percibe su ambiente mediante sensores y que responde o actúa en tal ambiente mediante efectores.

Atendiendo a estos planteamientos se puede definir al agente como una entidad que está en algún ambiente y ejecuta acciones dependiendo del estado del ambiente.

De gran interés es asociar el término agente dentro de la visión de la inteligencia artificial con el desarrollo de este capítulo. De acuerdo a lo comentado anteriormente, la IA puede ser enfocada desde la perspectiva del desarrollo de agentes inteligentes. Esta idea, la cual puede ser considerada como un nuevo reto a corto plazo, está siendo avalada y respaldada por numerosos investigadores en la materia. Esta frase fue pronunciada por el Dr. Nicholas Jennings en su discurso al recoger el premio al mejor investigador novel del congreso internacional de Inteligencia Artificial celebrado en Estocolmo en 1999 (IJCAI): “Los agentes constituyen el próximo avance más significativo en el desarrollo de sistemas y pueden ser considerados como la nueva revolución en el software”. (Julián and Botti 2000)
Los científicos se preocupan por el desarrollo de nuevas teorías que les permitan a los ingenieros crear técnicas y herramientas de apoyo para una vida más sencilla y práctica. En el área computacional, los expertos se están dedicando al diseño, creación e implementación de sistemas inteligentes (Mancilla 2008), que permitan una mejor interrelación con el usuario. Los avances propuestos por las ciencias computacionales son

muy útiles ya que al relacionarse con otras áreas proveen la simplificación de algunas actividades.

Una rama de las ciencias computacionales es la Inteligencia Artificial, y dentro de ella se encuentra el área de Agentes Inteligentes, que está dedicada a la creación de sistemas que permitan la optimización de actividades humanas así como emular el comportamiento humano aportando un grado más de inteligencia a la experticia de sistemas que están simplificando y optimizando actividades en las cuales no habían sido capaces de implementarse.

De momento, se puede asegurar que, hoy en día, existe una idea generalizada dentro de la IA (Hípola, Vargas et al. 1999) de que nos encontramos ante el nacimiento de una nueva tecnología, la de agentes inteligentes, que permite abordar de una manera más apropiada la construcción de sistemas inteligentes más complejos aplicados a muy diversos campos. A continuación se describirá con más detalles lo que es esta nueva tecnología a la que se ha orientado la IA.

Se puede definir al agente inteligente como una entidad software que, basándose en su propio conocimiento, realiza un conjunto de operaciones destinadas a satisfacer las necesidades de un usuario o de otro programa, bien por iniciativa propia o porque alguno de éstos se lo requiere. (Hípola, Vargas et al. 1999)

Se considera a los Agentes Inteligentes como una pieza de software que ejecuta una tarea dada utilizando información recolectada del ambiente, para actuar de manera apropiada hasta completar la tarea de manera exitosa. El software debe ser capaz de auto ajustarse basándose en los cambios que ocurren en su ambiente de forma tal que un cambio en las circunstancias producirá un resultado esperado. (Gilbert 1995)

Este trabajo se apoya en la siguiente caracterización de un agente (Wooldridge and Jennings 1995) como aquel sistema informático que satisface las siguientes propiedades (las cuatro primeras se consideran básicas):

- ***Autonomía*:** tiene la capacidad de actuar sin intervención humana directa o de otros agentes.
- ***Sociabilidad*:** capacidad de interaccionar con otros agentes, utilizando como medio algún lenguaje de comunicación entre agentes.

- ***Reactividad*:** un agente está inmerso en un determinado entorno del que percibe estímulos y ante los que debe reaccionar en un tiempo preestablecido.
- ***Iniciativa*:** un agente no sólo debe reaccionar a los cambios que se produzcan en su entorno, sino que tiene que tener un carácter emprendedor y tomar la iniciativa para actuar guiado por los objetivos que debe de satisfacer.
- ***Movilidad*:** habilidad de un agente de trasladarse en una red de comunicación informática.
- ***Veracidad*:** propiedad por la que un agente no comunica información falsa intencionadamente.
- ***Benevolencia*:** un agente no tiene objetivos contradictorios y siempre intenta realizar la tarea que se le solicita.
- ***Racionalidad*:** un agente tiene unos objetivos específicos y siempre intenta llevarlos a cabo.

Teniendo en cuenta las anteriores definiciones se puede considerar al agente inteligente como una entidad software que, basándose en su propio conocimiento realiza un conjunto de operaciones para satisfacer una serie de necesidades de manera exitosa y cumple básicamente con las propiedades de autonomía, sociabilidad, reactividad, iniciativa y movilidad.

Los Agentes Inteligentes son racionales, es decir, hacen lo correcto, esto es lo que le permite al agente obtener el mejor desempeño. Para evaluar el desempeño es necesario definir qué tan exitoso ha sido un agente en la consecución de los objetivos para el cual fue programado (Mancilla 2008). Es necesario que el tiempo empleado en la realización de la tarea que sea aceptable, dependiendo del ambiente de acción. Los Agentes Inteligentes proporcionan una técnica para resolver problemas actuando en representación del usuario, para realizar diversas tareas tales como, búsqueda y filtraje de información, automatización de tareas, aplicaciones en la industria, etc. Y en la actualidad son objeto de estudio hasta en disciplinas como la psicología, sociología y en algunas otras ramas de las ciencias sociales.

Los estudios de Agentes Inteligentes comienzan cuando se adquiere la capacidad de representar simbólicamente aspectos del mundo real, (*Physcal Symbol System Hypotesis*), para tener un comportamiento inteligente, que se identifica con la utilización del

razonamiento lógico explícito para decir que es lo que se tiene que hacer. Entre 1984 y 1985, surgen problemas con el razonamiento simbólico creando así los primeros agentes reactivos; y es hasta 1990 cuando comienzan a aparecer algunas arquitecturas híbridas estratificadas por capas, que simulan los primero agentes inteligentes (Mancilla 2008).General Magic en 1990 fue la primera compañía en invertir en tecnología de agentes para construir un sistema basado en agentes para dispositivos portátiles. Fue llamada "Telscrip" y lo utilizó Motorola y AT&T.

Se creó el lenguaje de manipulación y consultas de conocimiento KQLM (*Knowledge Query and Manipulation Language*), que funciona para compartir conocimiento y comunicación entre los mismos agentes. En 1995, con Windows 95 y las interfaces gráficas, nace Microsoft Agent. Posteriormente Genesereth propone los agentes como un medio de integración del software. Los agentes encapsulan los programas y mediante la definición de unas primitivas, permiten el intercambio de órdenes y datos entre los programas (Genesereth 1992). Otro es un lenguaje de representación del conocimiento denominado KIF (Formato de Intercambio del Conocimiento; Knowledge Interchange Format) (Mancilla 2008).

A continuación se citan las características más importantes de los Agentes Inteligentes según (Russell and Norvig 1996), de varias de estas no se harán mención pues ya están comentadas en el concepto de Wooldridge:

- **Pro actividad:** los agentes pueden reaccionar por iniciativa propia sin necesidad de que el usuario tenga que activarlo.
- **Orientación hacia el objeto final.** Divide su tarea compleja en varias actividades pequeñas para así poder lograr la meta compleja.
- **Adaptabilidad.** El agente debe ser capaz de ajustarse a los hábitos, formas de trabajo y necesidades del usuario.
- **Colaboración**. El agente debe ser capaz de determinar información importante ya que el usuario puede proporcionar información ambigua.

A continuación se describirá, de manera general, una clasificación de los Agentes Inteligentes en base la relación existente entre percepciones y acciones (Mancilla 2008), la aplicación a la que sirven, y características especiales.

1.2.1 Clasificación dependiendo de la relación entre percepciones y acciones:

Agentes de Reflejo Simple: Actúa encontrando una regla cuya condición coincida con la situación actual (definida por la percepción) y efectuando la acción que corresponda a tal regla.

Agentes Bien Informados de todo lo que Pasa: Actualiza constantemente la información que le permita discernir entre estados del mundo y su evolución; además de necesitar conocer como las acciones del propio agente están afectando al mundo; así se mantiene informado acerca de esas partes no visibles de él.

Agentes Basados en Metas: Es sencillo cuando con una sola meta se alcanza una acción. Debe ser flexible con respecto a dirigirse a diferentes destinos, ya que al marcar un nuevo destino, se crea en el agente una nueva conducta.

Agentes Basados en Utilidad: La utilidad es una función que correlaciona un estado y un número real mediante el cual se caracteriza el correspondiente grado de satisfacción.

1.2.2 Clasificación de acuerdo al tipo de aplicación

Agente de interfaz o usuario: Funciona como un asistente personal, sus características principales son: la autonomía y el aprendizaje. Enseñan al usuario a utilizar una aplicación en particular, poseen una base de conocimiento donde almacena el conocimiento adquirido por el usuario o por otros agentes

Agentes de Búsqueda: No son simplemente técnicas de búsqueda, sino que tienen que interpretar patrones de búsqueda. Debe ser capaz de crear información útil para el usuario a partir de pedazos de información.

Agentes de Monitoreo: Estos avisan a los agentes de interfaz sobre algún cambio en el contenido de alguna página Web.

1.2.3 Clasificación de acuerdo a características especiales

Agentes Deliberantes o Proactivos: Son agentes que poseen mucho conocimiento del entorno en el que se encuentran y son capaces de crear nuevos planes y adelantarse a lo que va a ocurrir en su entorno. En esta clasificación encontramos el modelo BDI (Belief, Desire, Intention) (Bellifemine, Poggi et al. 2001).

Agentes Reactivos: Son sistemas estímulo-respuesta que actúan a partir de la observación directa y continua del entorno. Se adaptan perfectamente a los entornos dinámicos ya que no tienen que actualizar ninguna representación interna del entorno como los agentes BDI.

Agentes Estacionarios: Son un tipo de agente que no poseen la capacidad de desplazarse y salir del entrono.

Agentes Móviles: Son agentes que tienen la capacidad de desplazarse a través de una red; de esta forma cambian el entorno en el que se ejecutan. Se reduce el consumo de recursos en la máquina en la que se encontraba inicialmente el agente.

1.2.4 Estructura de los Agentes inteligentes

Para realizar una correcta implementación de los agentes inteligentes es necesario conocer su estructura interna. Ésta contiene, como muestra el siguiente esquema, dos partes fundamentales, la arquitectura y el programa de agente (Mancilla 2008) obedeciendo a la ecuación:

Agente = Programa de Agente + Arquitectura.

La arquitectura se refiere la parte Hardware (sensores y efectores) que están entre el entorno y el Programa de Agente; proporciona al programa las percepciones del entorno que se obtienen a partir de los sensores e indica a los efectores los comandos asociados a las acciones que tienen que ejecutar. Para ejemplificarla podemos mencionar una computadora, una cámara, un brazo robótico u otro dispositivo que funcione como sensor.

El programa de agente es la parte Software que se encarga de procesar las percepciones obtenidas a partir de los sensores para determinar qué acciones deben ejecutar los efectores. Procesa la información que proviene de los sensores para determinar la acción más adecuada en cada momento.

Hablando en términos de programación, existe también un programa esqueleto de agente, en el que de manera general se establece las funciones que realizará el agente inteligente de manera interna (Russell and Norvig 1996).

Dentro del entorno computacional se pueden identificar varias aplicaciones de los agentes inteligentes como la administración de redes y sistemas, acceso remoto y administración, correo y mensajería, recuperación y manejo de información, colaboración, workflow y gestión administrativa, comercio electrónico, interfaces de usuario adaptativas, en la medicina y aplicaciones industriales (Mancilla 2008).

Todos estos avances de la tecnología han favorecido al desarrollo de la inteligencia artificial distribuida (IAD) se define como "aquella parte de la IA que se centra en comportamientos inteligentes colectivos que son producto de la cooperación de diversos agentes" (Choque 2009).

1.3. Sistemas Multiagentes.

Un sistema multiagente (SMA) se compone de dos o más agentes relacionados. Un agente es un sistema computacional autónomo y flexible, que es capaz de actuar en un entorno(Wooldridge and Jennings 1995).

La sociedad de agentes está organizada como una red en la cual los nodos representan dichas entidades, y los enlaces los puentes de intercambio de información entre los mismos. A partir de la organización de los enlaces, el intercambio de información puede hacerse o bien directamente, o bien indirectamente a través de un intermediario.

A partir del control esbozado en el párrafo anterior, se puede decir que existen dos tipos fundamentales de sociedades en la Inteligencia Artificial Distribuida: la centralizada y la descentralizada (Choque 2009).

Los Sistemas Multiagente (SMA) se preocupan por coordinar la conducta inteligente de agentes autónomos. Estos agentes forman parte de una colección y pueden coordinar su conocimiento, objetivos, habilidades y planes juntamente para tomar una acción o resolver una meta global, estos pueden tener conocimiento parcial sobre el problema y las soluciones. En estos sistemas debe haber un proceso de racionalización para la coordinación del conjunto de agentes. Por lo general en estos sistemas MA los agentes con sus creencias, deseos e intenciones construyen el problema y el plan o secuencia de acciones para solucionarlo.

El proceso de generación de un plan es llamado planeación, como ya se vio los planes en la IA son vistos como una secuencia de acciones de un agente, de tal manera que es capaz de cambiar su entorno y lograr uno o más objetivos (Choque 2009).

En resumen, los SMA son una tecnología software general motivada en cuestiones fundamentales de investigación acerca de autonomía, cooperación, formación de grupo, etc. Se ocupa de responder preguntas tales como "¿Qué puede ser hecho?", "¿Cómo puede ser

realizado?" y se aplica a una gran variedad de dominios: comercio electrónico, control inteligente de producción, robótica, recuperación de la información, etc.

La comunicación establecida por dos o más agentes puede ser vista como los procesos que componen el episodio de comunicación, protocolos utilizados en la comunicación, arquitecturas de comunicación y tipos de comunicación.

1.4. Lenguajes de programación

Sin lugar a dudas una gran necesidad que se evidencia es la programación de dichos agentes, para satisfacerla se puede emplear lenguajes de agentes.

Un lenguaje de agentes es un sistema que permite programar sistemas de computación en términos de algunos conceptos desarrollados por la teoría de agente. Se espera que aunque sea haya una estructura que represente a un agente, aunque se debería esperar que dicho lenguaje tuviera más conceptos de la teoría de agentes como creencias, metas, objetivos, planificación etc. (Choque 2009).

El interés por los lenguajes de agentes surgió de la propuesta de Shoham para crear un nuevo paradigma de programación orientada a agentes. El lenguaje propuesto fue el Agent0, en este la principal entidad es el agente y su estado es un conjunto de creencias, habilidades, acciones y objetivos. También existen métodos para intercambio de mensajes, solicitud y asignación de tareas.

Otra alternativa son los enfoques orientados a teorías de agentes, en donde se enuncia definiciones de los SMA para luego tratar estas como implementaciones. Un lenguaje ejemplo de esta alternativa es ConCOLOG (Blanco 2002) en donde se modela la ejecución de tareas asignadas a varios agentes y como afectan al entorno. La implementación de estas características de un SMA se basa en entidades de conocimiento modificables por las tareas. El conjunto de aplicaciones que se pueden dar sobre las tareas son axiomas de precondición de tareas y axiomas de marco que afectan a la ejecución de tareas

El problema de estos tipos de lenguajes es que su desarrollo se hace para aplicaciones pequeñas y medianas, ante aplicaciones de complejidad mayor se deben utilizar lenguajes de alto nivel como C++ y Java, el problema de utilizar estos lenguajes es que no conciben una entidad agente como tal, claro está que hay plataformas como JADE (De Giacomo, Lespérance et al. 2000) que definen clases que representan entidades de agentes y métodos

que representan en muchos casos implementaciones de los temas de la Inteligencia Artificial Distribuida (Choque 2009)

En principio, cualquier lenguaje de programación puede ser utilizado. Siendo así de amplio el espectro en el cual se puede escoger un lenguaje para programar un sistema experto. Atendiendo a la forma de estructurar sus instrucciones según (Choque 2009), se les puede dividir en:

IMPERATIVOS: PASCAL, C/C++.

FUNCIONALES: LISP.

DECLARATIVOS: PROLOG, CHIP, OPS5.

ORIENTADOS A OBJETOS: SmallTalk, Hypercard, CLOS, Python.

Tradicionalmente LISP y PROLOG han sido los lenguajes que se han utilizado para la programación de sistemas expertos.

i). LISP

Su nombre se deriva de LISt Processor. LISP fue el primer lenguaje para procesamiento simbólico. John McCarthy lo desarrolló en 1958, en el Instituto de Tecnología de Massachusetts (MIT), inicialmente como un lenguaje de programación con el cual los investigadores pudieran implementar eficientemente programas de computadora capaces de razonar. Rápidamente LISP se hizo popular por su capacidad de manipular símbolos y fue escogido para el desarrollo de muchos sistemas de Inteligencia Artificial.

ii). PROLOG

PROgramming in LOGic (PROLOG), es otro de los lenguajes de programación ampliamente utilizados en IA. PROLOG fue desarrollado en Francia, en 1973 por Alain Colmenauer y su equipo de investigación en la Universidad de Marseilles.

Inicialmente fue utilizado para el procesamiento de lenguaje natural, pero posteriormente se popularizó entre los desarrolladores de aplicaciones de IA por su capacidad de manipulación simbólica. Utilizando los resultados del grupo francés, Robert Kowalski de la Universidad de Edimburgo, en Escocia, desarrolló la teoría de la programación lógica. La sintaxis propuesta por Edimburgo, se considera el estándar de facto del PROLOG (Choque 2009).

Con el avance de estas tecnologías indiscutiblemente que los agentes pueden utilizarse en la identificación de procesos en las industrias logrando un gran paso de avance en la automatización de las mismas.

1.5. Identificación de sistemas.

Un sistema es toda realidad en la que interactúan variables de diferentes tipos para producir señales observables. Las señales observables que son de interés para el observador se denominan salidas del sistema, mientras que las señales que pueden ser manipuladas libremente por dicho observador son las entradas del mismo. El resto de señales que influyen en la evolución de las salidas pero no pueden ser manipuladas por el observador se denominan perturbaciones (López 2009).

Cuando se necesita conocer el comportamiento de un sistema con determinadas entradas y condiciones en muchos casos la experimentación resulta muy complejo, por lo que es necesario trabajar con alguna representación que se aproxime a la realidad, la cual se define como modelo.

Básicamente, un modelo es una herramienta que permite predecir el comportamiento de un sistema sin necesidad de experimentar sobre él (López 2009).

Existen varios tipos de modelos de acuerdo al grado de formalismo matemático que poseen, por ejemplo, los modelos mentales, intuitivos o verbales, modelos no paramétricos y los modelos paramétricos o matemáticos.

Los modelos paramétricos o matemáticos: Modelo desarrollado a partir de principios físicos, químicos, etc. Los parámetros describen la característica o personalidad del proceso (ganancia, tiempo muerto, constante de tiempo). Un 80% de los procesos químicos pueden ser modelados mediante estos parámetros G(s)=kp*e^-ts / s + 1 (Compuequipos 2009). Para aplicaciones más avanzadas, puede ser necesario utilizar modelos que describan las relaciones entre las variables del sistema mediante expresiones matemáticas como pueden ser ecuaciones diferenciales (para sistemas continuos) o en diferencias (para sistemas discretos).

Un método para obtener el modelo de un sistema es identificación de sistemas, a continuación se define de manera general:

Identificación de sistemas: Proceso de hallar una expresión que represente matemáticamente un sistema físico usando datos experimentales. (Compuequipos 2009)
El uso de datos reales para identificar los parámetros del modelo provee a éste de una gran exactitud, pero el proceso de identificación se ve tanto más facilitado cuanto mayor sea el conocimiento sobre las leyes físicas que rigen el proceso.

Identificación de Sistemas (IS) es un término usado por la comunidad de control, es un término más amplio que incluye la estructura del modelo y los parámetros correspondientes a ese modelo y los valores de los estados en cierto instante o como funciones de tiempo. Además incluye los métodos no-paramétricos.

1.5.1. Metodología para el proceso de identificación.

1. Obtención de datos de entrada – salida: Para ello se debe excitar el sistema mediante la aplicación de una señal de entrada y registrar la evolución de sus entradas y salidas durante un intervalo de tiempo.

2. Tratamiento previo de los datos registrados: Los datos registrados están generalmente acompañados de ruidos indeseados u otro tipo de imperfecciones que puede ser necesario corregir antes de iniciar la identificación del modelo. Se trata, por tanto, de 'preparar' los datos para facilitar y mejorar el proceso de identificación.

3. Elección de la estructura del modelo: Si el modelo que se desea obtener es un modelo paramétrico, el primer paso es determinar la estructura deseada para dicho modelo. Este punto se facilita en gran medida si se tiene un cierto conocimiento sobre las leyes físicas que rigen el proceso.

4. Obtención de los parámetros del modelo. A continuación se procede a la estimación de los parámetros de la estructura que mejor ajustan la respuesta del modelo a los datos de entrada-salida obtenidos experimentalmente.

5. Validación del modelo. El último paso consiste en determinar si el modelo obtenido

satisface el grado de exactitud requerido para la aplicación en cuestión. Si se llega a la conclusión de que el modelo no es válido, se deben revisar los siguientes aspectos como posibles causas:

a) El conjunto de datos de entrada-salida no proporciona suficiente información sobre la dinámica del sistema.

b) La estructura escogida no es capaz de proporcionar una buena descripción del

modelo.

c) El criterio de ajuste de parámetros seleccionado no es el más adecuado.

Dependiendo de la causa estimada, deberá repetirse el proceso de identificación desde el punto correspondiente. Por tanto, el proceso de identificación es un proceso iterativo (López 2009).

El objeto de la identificación de sistemas es encontrar una representación del sistema de la forma y(t)=G(q)u(t)+e(t) y diseñar perturbaciones sobre u(t) para ser aplicadas al proceso y obtener suficiente información que permita identificarlo (Compuequipos 2009).

La forma directa de identificar un proceso es obteniendo los parámetros del modelo dinámico de la respuesta a un cambio en la entrada.

1.5.2. Clasificaciones.

- Dependiendo del tipo de modelo obtenido (López 2009):

1. Métodos no paramétricos, que permiten obtener modelos no paramétricos del sistema bajo estudio. Algunos de estos métodos son: análisis de la respuesta transitoria, análisis de la respuesta en frecuencia, análisis de la correlación, análisis espectral, análisis de Fourier, etc.

2. Métodos paramétricos, que permiten obtener modelos paramétricos. Estos métodos requieren la elección de una posible estructura del modelo, de un criterio de ajuste de parámetros, y por último de la estimación de los parámetros que mejor ajustan el modelo a los datos experimentales.

1.5.3. Métodos empleados en la identificación

- Métodos no-paramétricos

Entre los métodos no paramétricos se pueden mencionar Análisis de Transitorios, Análisis de Frecuencia y Análisis de Correlación, estos no se explicaran pues no son de interés para el desarrollo del presente trabajo.

La identificación no paramétrica cuenta con las ventajas de no necesitar un procesamiento complejo de los datos, ni estudio previo sobre el proceso o la planta y permite centrar los datos que se obtienen alrededor de frecuencias establecidas sin embargo el modelo obtenido no puede usarse de forma directa para la simulación

Antes de explicar los métodos paramétricos es necesario señalar que los modelos paramétricos se describen en el dominio discreto pues los datos para la identificación se obtienen por muestreo, aunque si se necesita un modelo continuo, siempre una transformación del dominio discreto al continuo es posible.

- Métodos paramétricos

1. Modelo ARX (AutoRegressive with eXternal input):

Son la primera elección en un procedimiento de identificación de sistemas lineales.

2. Modelo ARMAX (Auto Regressive Moving Average):

Describe el error en la ecuación como un promedio móvil (Moving Average).

3. Modelo OE (Output Error):

Es un modelo ARMAX con relación in/out sin perturbación más ruido blanco aditivo en la salida.

4. Modelo BJ (Box-Jenkins):

Es una generalización del modelo output error.

1.5.4. Métodos para el ajuste de parámetros

Se basan en la predicción del error (López 2009) (la diferencia entre la salida del proceso y la predicción hecha por el modelo).

1. Errores de predicción o residuos de un modelo

Todo modelo matemático es capaz de predecir el valor de la salida del sistema en función de las entradas y salidas en instantes anteriores. Se llama error de predicción e(t,q)

a la diferencia entre la salida estimada por el modelo y la salida real del sistema en un determinado instante de tiempo.

2. Regresión lineal.
3. Método de mínimos cuadrados.
4. Existencias de herramientas software que proporcionan diferentes algoritmos para el ajuste de parámetros.

Cabe destacar que esta etapa del proceso de identificación se facilita por la existencia de estas herramientas, en este caso la más significativa actualmente en la carrera es el Toolbox de Identificación de Matlab por diversas razones pues este tiene una gran utilidad práctica, dentro de otros aspectos permite la simulación de los sistemas además de que se ha venido trabajando con el Matlab desde los primeros años de la ingeniería.

1.5.5. Herramienta para la identificación.

En muchas ocasiones no se puede obtener teóricamente (mediante un modelo matemático) la función de transferencia de una planta, debido a diversos factores. En estos casos se puede hacer una identificación del sistema a partir de mediciones hechas a la entrada y salida de la planta. Para este efecto, el programa MATLAB tiene una herramienta llamada SYSTEM IDENTIFICATION (BARAHONA 2011). La herramienta tiene una serie de funciones programadas que sirven para realizar la siguiente secuencia de operaciones:

- Carga archivo de datos, es decir los datos registrados en un osciloscopio.

- Organiza y filtra los datos, removiendo los valores medios y filtrando el ruido en los mismos. Adicionalmente separa datos para identificación y datos para validación.

- Aplica métodos para la predicción del error, mediante un modelo del sistema en variables de estado.

- Realiza proceso de validación, comparando gráficamente los resultados. Adicionalmente, entrega el porcentaje de aciertos y el análisis de correlación de los residuos con la entrada.(Botero and Ramírez 2008)

- Funciones para simulación y predicción
  - IDINPUT: Genera los datos de entrada para propósitos de simulación.
  - IDSIM: Simula un sistema lineal general.

- ❖ PREDICT: Calcula las predicciones de la salida del modelo.

➢ Funciones para manipulación de datos

- ❖ DTREND: Sirve para remover tendencias.
- ❖ IDFILT: Para realizar filtros a los datos del proceso.

➢ Funciones para Estimación Paramétrica

- ❖ AR: Estima un modelo AR.
- ❖ ARX: Estima un modelo ARX.
- ❖ ARMAX: Estima un modelo ARMAX.
- ❖ BJ: Estima un modelo BOX JENKINS.
- ❖ IV4: Estima un modelo ARX usando el método de la variable instrumental de cuatro etapas.
- ❖ CANSTART: Estima modelos multivariables en forma canónica de espacio de estado; generalmente usado junto con N4SID.
- ❖ N4SID: Estima modelos de espacio de estado usando un método de subespacio.
- ❖ PEM: Estima un modelo lineal general.

➢ Funciones para Validación

- ❖ COMPARE: Compara la salida simulada o predicha con la salida del modelo.
- ❖ PE: Calcula los errores de predicción.

1.6. Tratamiento de residuales

Generalmente la industria trata o dispone separadamente los residuos líquidos industriales de los domésticos (producidos en baños y cocina de la planta), que es como la normativa ambiental, en general, lo indica (Guzmán, Karin et al. 1998).

Para la depuración del agua residual actualmente existen diversos métodos, entre los que se pueden mencionar al pretratamiento, tratamiento primario, secundario y terciario (Manrique 2003), los cuales se complementan en dependencia del grado de contaminación del agua a tratar. El pretratamiento consiste en la eliminación de sólidos gruesos, suspendidos, grasas/aceites, que puedan ocasionar problemas de mantenimiento y funcionamiento de los procesos posteriores; el tratamiento primario reduce alrededor del 30% de la materia orgánica y un 60% de sólidos suspendidos, en el tratamiento secundario se elimina el resto

de los compuestos biodegradables y sólidos suspendidos, mediante procesos físicoquímicos y biológicos (lodos activados, lagunas aireadas, zanjas de oxidación, filtros rociadores, biodiscos o combinados) donde las bacterias son el principal agente purificador en forma de "biomasa". El tratamiento terciario o avanzado, se encarga de la remoción de nutrientes para prevenir eutrofización de las fuentes receptoras, o bien para incrementar la calidad de un efluente secundario para un rehúso adecuado.

Como se expresó anteriormente los tratamientos biológicos forman parte del tratamiento secundario. Estos consisten en buscar la manera de biodegradar naturalmente los materiales disueltos o en suspensión en una mezcla. El factor de diseño más importante está en determinar el tratamiento que permita la biodegradación más rápidamente, este factor depende del tipo de microorganismo empleado, los nutrientes que le van a permitir desarrollarse a dicho microorganismo, y el lugar físico donde se va a desarrollar el tratamiento (tipo de suelo, pH, temperatura del ambiente, condiciones climáticas, oxigenación natural, etc.). Dado que estos microorganismos son sensibles a cambios en su ecosistema, se debe mantener dichos parámetros en niveles constantes, además se debe asegurar que no ingresen sustancias que resulten tóxicas a los microorganismos (Mendiburu 2003).

El tratamiento biológico implica una compleja interacción de diversas especies microbianas. La biodegradación de compuestos orgánicos se basa en la oxidación biológica de los mismos por la acción de microorganismos. (Ercoli 1998).

En una mezcla de poblaciones microbianas son muy importantes aquellos organismos que pueden iniciar procesos catabólicos, factor que unido a la resistencia de varios compuestos contribuyen a que la velocidad a la que los microorganismos oxidan hidrocarburos varíe marcadamente.

Los mecanismos bioquímicos de la degradación son complejos y dependen de las condiciones fisicoquímicas del medio, del tipo de sustrato y del microorganismo. Dichos mecanismos no están totalmente estudiados y sólo se ha investigado la degradación de algunos compuestos específicos (Ercoli 1998). Cada especie de microorganismos tiene una capacidad específica para degradar hidrocarburos: solo ataca algunos compuestos específicos y en un grado determinado. Cuando se tiene una mezcla compleja como puede ser un lodo de refinería se recurre al uso combinado de un grupo de especies degradadoras.

Estos conjuntos de microorganismos se denominan consorcios microbianos. Como se aprecia de manera general la biodegradación acontece bajo un amplio rango de condiciones como características del contaminante en cuanto a ph y grado de contaminación por solo citar algunas: humedad, población de microorganismos, etc. Cuando se logra cubrir las especifidades dichas se logran resultados muy favorables, sin embargo las afirmaciones anteriores evidencian la dinámica compleja que presentan los tratamientos biológicos, la mayoría de esos bioprocesos son no lineales, factores que exigen evidentemente varias prestaciones de los controladores y aumentan el grado de complejidad de los sistemas de control para estos procesos.

Conclusiones parciales

Dada la complejidad de los procesos biotecnológicos de tratamientos de residuales acentuada en la dinámica no lineal que poseen la mayoría, la no existencia de un método general para afrontar todos los sistemas no lineales, además de varias especificaciones que exigen un cierto grado de autonomía en el controlador, en el sentido de capacidad de reacción y replanteamiento de objetivos ante fallos en el sistema los agentes inteligentes poseen las características tales como autonomía, sociabilidad, la iniciativa y racionalidad necesarias para desarrollar la implementación en el área de identificación de sistemas en procesos de tratamientos biológicos para residuales.

La investigación será un gran paso de avance en el uso de los agentes inteligentes en aplicaciones industriales, sobre todo en el sector petrolero pues como se ha corroborado a pesar de los adelantos logrados en disímiles aplicaciones de los agentes, en el área de la identificación de procesos la aplicación de la tecnología de SMA no ha sido tan frecuente. (González, Hamilton et al. 2006).

CAPÍTULO 2. METODOLOGÍAS EMPLEADAS EN EL TRABAJO CON SISTEMAS MULTI-AGENTES.

Desde un punto de vista metodológico, el principal problema de los lenguajes de agentes es su aplicación a desarrollos de complejidad media. Los lenguajes de agentes pueden verse como lenguajes de implementación de más alto nivel que otros más convencionales como C++ o JAVA, pero con una particularidad: son muy pobres en mecanismos de abstracción y encapsulación (Gómez 2002). Los programas o especificaciones generados con ellos tienen como unidad principal de encapsulación el agente y de abstracción procedimental la tarea, lo cual limita su aplicación. Así pues, a los problemas de desarrollar un sistema directamente con un lenguaje de implementación se une la falta de medios para realizar esta tarea incrementalmente.

Entonces, la necesidad de metodologías es doblemente justificable. Por un lado, para cubrir las deficiencias de los lenguajes de agentes en cuanto a mecanismos de encapsulación y abstracción, y por otro lado, para facilitar la comprensión de sistemas complejos tal y como ocurre con los lenguajes convencionales de implementación.

2.1 Metodologías para el diseño de Sistemas Multiagente (SMA).

Después de una amplia revisión bibliografía en el tema de los sistemas multiagentes, se ha podido identificar una serie de metodologías que han sido utilizadas por diferentes autores a la hora de trabajar con SMA.

De entre las metodologías existentes, se ha seleccionado un conjunto utilizando tres criterios de selección propuestos por (Gómez 2003), ellos son:

- Utilización de diferentes vistas para la especificación del sistema.
- Incorporar la idea de proceso de desarrollo.

- Integrar de técnicas de ingeniería y teoría de agentes.

De acuerdo con estos criterios, se han identificado una serie de metodologías. Antes de pasar a analizar con más detalles dichas metodologías, se presentarán algunas de sus características generales.

- Ingeniería de vocales (*vowel engineering*) (Demazeau 1995): Fue una de las primeras en considerar diferentes aspectos como agentes, entorno, interacciones y organización en el desarrollo de SMA (Gómez 2003).
- MAS-CommonKADS (Iglesias 1998): Es una extensión de la metodología *CommonKADS*, añade los aspectos que son relevantes para los sistemas multiagente (MAS).
- El diseño basado en BDI (Kinny, Georgeff et al. 1997): Ha influido notablemente en la forma de concebir el control de los agentes.
- ZEUS (Nwana, Ndumu et al. 1999). De gran utilidad a la hora de implementar. Combina los distintos resultados de investigación en agentes en un sistema ya ejecutable.
- MaSE (DeLoach 2001). Se aprende rápido, solo hay que experimentar con la herramienta. Se compone básicamente de dos fases: análisis y diseño.
- GAIA (Wooldridge, Jennings et al. 2000). De gran influencia, que estudia la definición de vistas en una metodología y trata de integrarse en un ciclo de vida de software tipo cascada.
- INGENIAS(Gómez 2002) (Gómez and Fuentes 2002). Se considera como evolución de las ideas de MESSAGE, incorpora nuevas herramientas de soporte, define un conjunto de meta-modelos con los que hay que describir el sistema.
- RT-MESSAGE (Julián and Botti 2003). Orientada a la construcción de Sistemas Multiagente de Tiempo Real, basándose en la metodología MESSAGE.
- PROHA(Cenjor and García 2005). Construida a partir de PROSA. El proceso de diseño se basa en el planteamiento de un sistema heterárquico de agentes deliberativos, en concreto, de agentes BDI.

2.1.1 Metodología Vowel Engineering (ingeniería de voces).

Se trata de la metodología seguida en el grupo MAGMA (MAGMA 2003). El término *vowel engineering* viene de que el sistema final depende de la ordenación y agrupamiento de aspectos identificados por cuatro vocales: A (por agentes), E (por entorno), I (por interacciones) y O (por organización). Para cada aspecto, se utilizan componentes y técnicas específicas. Para representar agentes se usa desde simples autómatas hasta complejos sistemas basados en conocimiento. La forma de ver las interacciones van desde modelos físicos (propagación de onda en el medio físico) hasta los actos del habla (*speech acts*). Las organizaciones van desde aquellas inspiradas en modelos biológicos hasta las gobernadas por leyes sociales basadas en modelos sociológicos (Gómez 2003).

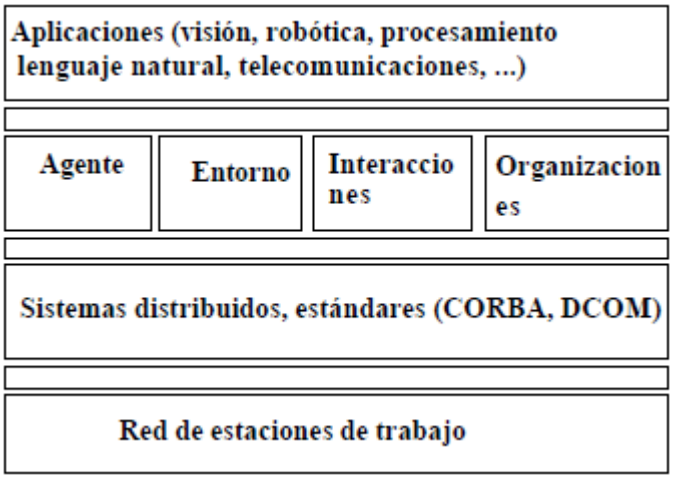


Fig. 2.1 Arquitectura de capas del sistema Multiagente en Vowel Enginering

El propósito de *Vowel engineering* es lograr librerías de componentes que den soluciones al diseño, para que posteriormente, el diseñador seleccione un modelo de agente, un modelo de entorno, un modelo de interacciones y modelos de organización a instanciar. En cuanto al aspecto de interacción, se aprecia un lenguaje, para la descripción de protocolos de interacción basados en procesos de comunicación síncronos o asíncronos, donde la semántica es muy similar a la de los actos del habla. La representación en sí se hace mediante redes de transición en las que los arcos se corresponden con los mensajes intercambiados y los estados reflejan la situación global (no hay posibilidad de que un estado se refiera al estado de un agente concreto). El proceso de desarrollo consiste en tener en cuenta los elementos señalados por las vocales en un cierto orden. El orden se decide en

función del tipo de sistema que queramos tener. Por ejemplo, si se empieza por la vocal O, se tendrá un sistema en el que las relaciones sociales son lo más importante. Si se empieza por la A, se tendrá un sistema en el que probablemente la organización surja como resultado de la interacción de agentes aislados.

Acerca de *Vowel Engineering* (Ricordel 2001) propone la implementación mediante la plataforma *Volcano*. La plataforma utiliza el ensamblaje de componentes utilizando lenguajes de composición de arquitecturas, concretamente UniCon (Shaw, DeLine et al. 1995), este fue para ganar experiencia con detalles prácticos de un lenguaje de descripción, enfatiza los aspectos estructurales de las arquitecturas de software, entre sus objetivos estaban resolver problemas reales de dirección de la descripción y composición del sistema y proveer un prototipo de herramienta práctica, soporta notaciones gráficas y textuales con intercambio entre las representaciones. También soporta herramientas de análisis y notaciones de especificaciones desarrollados por otros, preservando la compatibilidad con herramientas de programación de uso común. El desarrollo consiste en ensamblar componentes que pueden proceder de una librería o ser desarrolladas *ad-hoc* (Muñoz 2009), esta última es una red inalámbrica descentralizada, la red es ad hoc porque cada nodo está preparado para reenviar datos a los demás. Cada componente pertenece a una categoría concreta de las cuatro consideradas. *Vowel Engineering* ha sido una de las primeras metodologías en modelar sistemas a partir de la combinación de diferentes aspectos. La combinación de aspectos se indicaba inicialmente en forma de instrucciones en lenguaje natural, pero en los últimos trabajos ha evolucionado hacia la utilización de lenguajes de arquitecturas. Como consecuencia de esta forma de desarrollo, se facilita la reutilización de código. Aunque es prometedor, el trabajo en *Vowel Engineering* aún no está completo ya que faltan herramientas de soporte. El proceso de desarrollo es el punto débil de esta metodología (Gómez 2003).

2.1.2 MAS-CommonKADS

Esta metodología se basa en CommonKADS (Tansley and Hayball 1993) aplicando ideas de metodologías orientadas a objetos para su aplicación a la producción de SMA (Iglesias 1998) (Iglesias , Mercedes et al. 1998). CommonKADS es una metodología diseñada para el desarrollo de Sistemas Basados en Conocimiento (SBC), logran una aproximación al

desarrollo de estos SBC mediante la construcción de un número de modelos interrelacionados que capturan los principales rasgos del sistema y de su entorno(Iglesias 1998). Hay seis modelos definidos en CommonKADS : *Modelo de la Organización (OM), Modelo de Tarea (TM), Modelo de Agente (AM), Modelo de Comunicaciones (CM), Modelo de la Experiencia (EM), Modelo de Diseño (DM).*

Se debe hacer notar que *CommonKADS* no fue diseñado para desarrollar sistemas multiagente por lo que se presentan una serie de problemas a la hora de aplicar directamente esta metodología al desarrollo de estos sistemas, dada su concepción, sin ninguna de las propiedades características de los agentes (proactividad, reactividad, colaboración, etc.), será necesario dar un nuevo enfoque a los modelos existentes. La concepción de agente dentro de *CommonKADS* es muy restrictiva.

Sin embargo CommonKADS se convirtió en el punto de partida de la metodología MAS-CommonKADS pues esta toma como marco de referencia los modelos identificados en CommonKADS e incorpora técnicas de modelado que facilitan su aplicación y que cubren los aspectos relevantes de un sistema multiagente. La definición de la metodología para desarrollo de sistemas multiagente se realiza, por tanto, desde una perspectiva integradora (Iglesias 1998).

MAS-CommonKADS ha sido la primera en plantear un desarrollo de SMA integrado con un ciclo de vida de software, concretamente el espiral dirigido por riesgos (Pressman 1982). Propone siete modelos para la definición del sistema (Iglesias 1998):

- *Modelo de Agente (AM):* específica las características de un agente: sus capacidades de razonamiento, habilidades, servicios, sensores, efectores, grupos de agentes a los que pertenece y clase de agente. Un agente puede ser un agente humano, software, o cualquier entidad capaz de emplear un lenguaje de comunicación de agentes.
- *Modelo de Organización (OM):* es una herramienta para analizar la organización humana en que el sistema multiagente va a ser introducido y para describir la organización de los agentes software y su relación con el entorno.
- *Modelo de Tareas (TM):* describe las tareas que los agentes pueden realizar: los objetivos de cada tarea, su descomposición, los ingredientes y los métodos de resolución de problemas para resolver cada objetivo.

- *Modelo de la Experiencia (EM):* describe el conocimiento necesitado por los agentes para alcanzar sus objetivos. Sigue la descomposición de *CommonKADS* y reutiliza las bibliotecas de tareas genéricas.
- *Modelo de Comunicación (CM):* describe las interacciones entre un agente humano y un agente software. Se centra en la consideración de factores humanos para dicha interacción.
- *Modelo de Coordinación (CoM):* describe las interacciones entre agentes software.
- *Modelo de Diseño (DM):* mientras que los otros cinco modelos tratan del análisis del sistema multiagente, este modelo se utiliza para describir la arquitectura y el diseño del sistema multiagente como paso previo a su implementación.

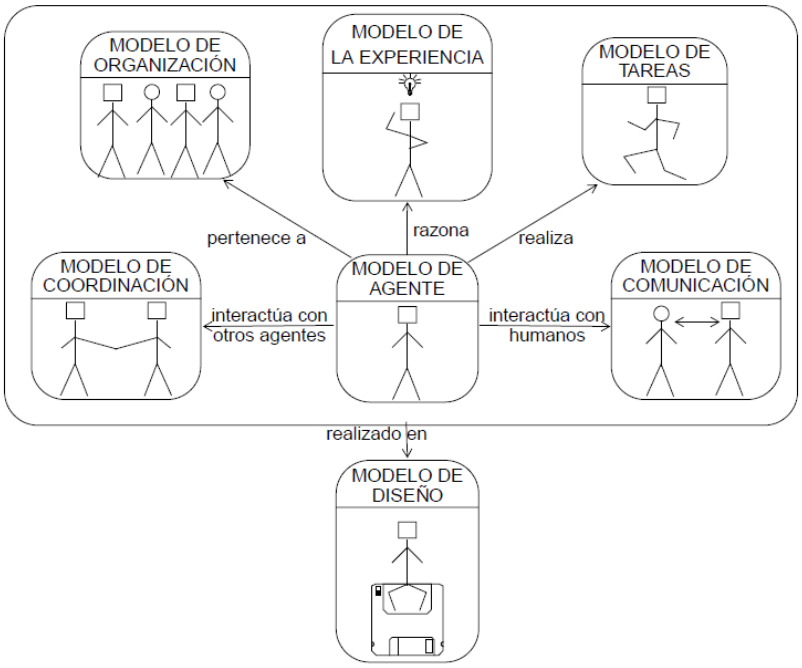


Fig. 2.2 Los modelos de MAS-CommonKADS

El modelo en sí parte de una descripción gráfica que luego se complementa con explicaciones en lenguaje natural de cada elemento. Existe por cada modelo una descripción de las dependencias respecto de otros modelos y de las actividades involucradas. Estos modelos se hayan descritos ampliamente por (Pressman 1982) en lenguaje natural, complementándose con otras notaciones como SDL (*Specification and Description Language*) (Union 2002) o MSC (*Message Sequence Chart*) (Union 2002) para

describir el comportamiento de los agentes cuando interaccionan. También se han desarrollado herramientas de soporte.

Esta metodología ha sido la primera en incorporar la idea de *proceso de ingeniería* en el sentido de Pressman (Pressman 1982), es decir, herramientas, métodos y procedimientos. De hecho, describe con bastante detalle cómo se debe definir el sistema teniendo en cuenta las dependencias entre los modelos.

El principal inconveniente de MAS-CommonKADS es que el nivel de detalle alcanzado en la descripción no es realizable sin el apoyo de herramientas de soporte. Lo que propone MAS-CommonKADS, entre otras cosas, no es una notación sino una lista detallada de elementos y relaciones a identificar en el sistema. Un desarrollador puede seguir la lista y generar la documentación requerida de forma manual, sin embargo el proceso es demasiado costoso y dado a errores.

En la versión actual de MAS-CommonKADS (Gómez 2003), la información que hay que obtener se expresa con lenguaje natural. A pesar del apoyo de una herramienta de soporte, al tener la especificación en lenguaje natural, se dificulta el análisis automático de la especificación generada. Así pues, para averiguar si existen elementos no utilizados o si existen contradicciones hay que revisar la documentación a mano. Para lograr lo mismo en MAS-CommonKADS habría que restringir el uso de lenguaje natural o bien incluir formalismos que logren una definición más precisa y menos ambigua del SMA.

A pesar de estos inconvenientes, MAS-CommonKADS constituye un ejemplo a seguir en lo que a metodologías se refiere. Es exhaustiva como pocas a la hora de detallar el sistema y además es consecuente con que el proceso de desarrollo en la mayoría de los casos es más complejo que un conjunto de pasos. De hecho, las ideas de MAS-CommonKADS han estado presentes en casi todos los trabajos referentes a metodologías desde hace años.

2.1.3 Metodología BDI (Belief, Desire, Intention)

Las arquitecturas BDI en español (Creencia, Deseo, Intención) se inspiran en un modelo cognitivo del ser humano (Bratman 1987). Según esta teoría, los agentes utilizan un modelo del mundo, una representación de cómo se les muestra el entorno. El agente recibe estímulos a través de sensores ubicados en el mundo. Estos estímulos modifican el modelo del mundo que tiene el agente representado por un conjunto de creencias, las cuales son

experiencias pasadas (Romero 2010). Para guiar sus acciones, el agente tiene *Deseos*. Un *deseo* es un estado que el agente quiere alcanzar a través de *intenciones*. Las intenciones son acciones a realizar para alcanzar un deseo, qué acción debe emprender en este momento (Romero 2010). Éstas son acciones especiales que pueden abortarse debido a cambios en el modelo del mundo. Aunque la formulación inicial es de Bratman, fueron Georgeff, Rao y Kinny (Kinny, Georgeff et al. 1997) quienes formalizaron este modelo y le dieron aspecto de metodología. De hecho es su trabajo, y no el original de Bratman, lo que aquí se comenta.

Con esta metodología los detalles esenciales de un sistema son capturados por tres diferentes tipos de modelos (Kinny, Georgeff et al. 1997):

1. Un Modelo Objeto captura información sobre los objetos dentro del sistema, describiendo sus estructuras de datos, las relaciones y las operaciones que ellos soportan.
2. Un Modelo Dinámico, este describe los estados, transiciones, eventos, acciones, actividades e interacciones que caracteriza el comportamiento del sistema.
3. Un Modelo Funcional, este describe el flujo de datos durante la actividad del sistema, de dentro y entre componentes del sistema.

Sin embargo para especificar el sistema de agentes, se emplean un conjunto de modelos que operan a dos niveles de abstracción: externo e interno. Primero, desde un punto de vista externo, un sistema se modela como una jerarquía de herencia de clases de agentes, de la que los agentes individuales son instancias. Las clases de agente se caracterizan por su propósito, sus responsabilidades, los servicios que ejecutan, la información acerca del mundo que necesitan y las interacciones externas. Segundo, desde un punto de vista interno, se emplean un conjunto de modelos (modelos internos) que permiten imponer una estructura sobre el estado de información y motivación de los agentes y las estructuras de control que determinan su comportamiento (creencias, objetivos y planes en este caso).

Desde el punto de vista externo la metodología para su elaboración y su refinamiento puede ser expresada como cuatro pasos principales (Kinny, Georgeff et al. 1997):

1. Identifique los roles del dominio de aplicación. Hay varias dimensiones en las cuales tal análisis puede ser emprendido; los roles pueden ser relacionados directamente a la aplicación o requeridos por la implementación del sistema.

2. Por cada rol, identifique sus responsabilidades asociadas y servicios que provee.

3. Por cada servicio, identifique las interacciones asociadas con la provisión de servicios. Identifique los eventos y condiciones a ser notadas.

4. Refine la jerarquía de agente. Donde hay información pública o servicios entre las clases de agente.

Para el modelado interno (punto de vista interno) la metodología puede ser expresada en dos pasos (Kinny, Georgeff et al. 1997):

1. Analice la manera de lograr las metas. Para cada meta analice los contextos diferentes en los cuales la meta tiene que ser lograda.

2. Construir las creencias del sistema. Analice varios contextos y condiciones que controlan la ejecución de actividades y acciones y descompóngalos por creencias.

En esta metodología, la integración con el ciclo de vida de software es reducida. Estos pasos se repiten haciendo que los modelos, que capturan los resultados del análisis, sean progresivamente elaborados, revisados y refinados.

Por ejemplo, el construir los planes y creencias de una clase de agente clarifica qué nivel de detalle se requiere en la representación del mundo que posee el agente.

Se imponen varias restricciones sobre la arquitectura que soporte estos modelos: que asegure que los eventos se responden en su momento, que las creencias se mantengan consistentemente, y que la selección de planes y ejecución se desarrolle de manera que refleje ciertas nociones de racionalidad.

Metodológicamente, la propuesta de Georgeff, Kinny y Rao es consecuente con la dificultad de generar los modelos que proponen. Como ellos admiten, existen dependencias entre los diferentes modelos que dificultan el proceso de generación de los modelos que describen el SMA. Como solución proponen un proceso iterativo e incremental (idea también reflejada en el Proceso Racional Unificado) (Jacobson, Rumbaugh et al. 1999) con

realimentaciones. Sin embargo, la forma en que tienen lugar estas realimentaciones no se llega a mostrar con detalle. Como en anteriores metodologías, se echa de menos la presencia de herramientas de soporte (Gómez 2003). Sin embargo, los modelos formales que se pueden encontrar tras esta metodología constituyen una referencia obligada para aquellos interesados en la integración de teorías de agentes y las prácticas de ingeniería.

2.1.4 Metodología ZEUS

ZEUS consta de una herramienta (Nwana, Ndumu et al. 1999) (ver anexo1a) y una metodología (Collis and Ndumu 1999), de forma similar a agenTool y MaSE. Desde su aparición, ZEUS se ha convertido en referencia de cómo debe ser una herramienta para el desarrollo de SMA. Sobre todo, por la forma en que combinan los distintos resultados de investigación en agentes (planificación, ontologías, asignación de responsabilidades, relaciones sociales entre agentes) en un sistema ya ejecutable. De hecho, la aplicación genera incluso programas para arrancar el sistema especificado e incluye herramientas de monitorización como el *Visor de Sociedad* que muestra los agentes existentes y sus relaciones, la *Herramienta de Control* para ver o modificar remotamente el estado de los agentes y los *Generadores de Informes* para obtener estadísticas de funcionamiento e informes de actuación de la sociedad de agentes.

La metodología ZEUS propone un desarrollo en cuatro etapas (Collis and Ndumu 1999): el análisis del dominio, el diseño de los agentes, la realización de los agentes y el soporte en tiempo de ejecución. Las etapas soportadas por la herramienta son la de *realización de los agentes* y la de *soporte en tiempo de ejecución*. Las etapas anteriores se basan en el uso de roles para analizar el dominio y en su asignación a agentes.

Entorno de desarrollo ZEUS

Los resultados a producir a lo largo del proceso de desarrollo son los siguientes:

- **Análisis del dominio.** Se orienta a obtener el modelo de roles. El modelo de roles se compone de diagramas UML de clases para representar roles, diagramas UML de colaboración para indicar qué mensajes se intercambian y fichas para describir los roles.

- **Diseño del agente.** Consiste en determinar qué necesita cada agente para poder desempeñar su cometido. Esto incluye revisar las tecnologías y disciplinas relacionadas con el diseño de agentes.
- **Implementación de los agentes y soporte en tiempo de ejecución.** Se trata de utilizar la herramienta ZEUS para trasladar los conceptos de diseño dentro de la plataforma que ZEUS tiene ya implementada. El resultado es un modelo ejecutable del sistema.

Hay poco que comentar acerca de ZEUS. Hasta la aparición de MaSE, ZEUS era herramienta referencia. Hoy en día se encuentran muchos más artículos referenciando MaSE que ZEUS (Gómez 2003). Ante la similitud de enfoques, se impone una breve comparativa entre ZEUS y MaSE.

Conceptualmente, ZEUS es superior a MaSE. Si bien la primera está más orientada a la aplicación de tecnología de agentes (planificación, definición de ontologías, secuenciación de tareas), la segunda se orienta más a las prácticas de ingeniería convencional (Gómez 2003).

Metodológicamente, ZEUS es más pobre que MaSE. El modelado de roles, propuesto en (Collis and Ndumu 1999), no profundiza en la aplicación de la herramienta dentro del proceso de desarrollo. El ámbito de la metodología se limita a estudiar cómo agrupar la funcionalidad del sistema dentro de cada rol, dejando aparte consideraciones acerca de cómo organizar las tareas, definir las ontologías y las dependencias sociales, aspectos que son modelables dentro de la herramienta.

El uso de la herramienta ZEUS no es trivial. Presenta tantas posibilidades que a un desarrollador con pocos conocimientos en agentes puede hacerle dudar. ¿Por dónde empezar? Por ello, se echa más en falta un proceso de desarrollo más detallado que indique en qué aspectos de la herramienta hay que concentrarse en cada momento (Gómez 2003).

Al estudiar ZEUS y MaSE nos podemos cuestionar si hay que tener una buena herramienta o una buena metodología. La opinión del autor es que ambas aún cuando deban ser independientes debe existir una herramienta lo suficientemente robusta como para poder implementar de forma eficiente las más altas potencialidades de una metodología bien fundamentada y definida.

2.1.5 Metodología MaSE

MaSE (Multiagent systems Software Engineering[1]) (DeLoach 2001) parte del paradigma orientado a objetos y asume que un agente es sólo una especialización de un objeto. La especialización consiste en que los agentes se coordinan unos con otros vía conversaciones y actúan proactivamente para alcanzar metas individuales y del sistema.

En MaSE los agentes son sólo una abstracción conveniente, que puede o no poseer inteligencia. En este sentido, los componentes inteligentes y no inteligentes se gestionan igualmente dentro del mismo armazón. El proceso de desarrollo en MaSE es un conjunto de pasos, la mayoría de los cuales se ejecutan dentro de la herramienta que soporta MaSE, AgentTool (DeLoach 2001) (ver anexo 1b).

La presente metodología (anexo 2a) se compone básicamente de dos fases: análisis y diseño (DeLoach 2001).

La fase de análisis en MaSE consta de tres pasos: capturar los objetivos, capturar los casos de uso y refinar roles. Como productos de estas etapas se esperan: diagramas de objetivos, que representan los requisitos funcionales del sistema; diagramas de roles, que identifica roles, tareas asociadas a roles y comunicaciones entre roles y entre tareas; y casos de uso, mostrados no como diagrama sino como una enumeración de los casos de uso considerados con la posibilidad de usar diagramas de secuencia para detallarlos.

La fase de diseño consta de cuatro pasos: crear clases de agentes, construir conversaciones, ensamblar clases de agentes y diseño del sistema. Como productos de estas etapas, MaSE espera: diagramas de clases de agentes, que enumeran los agentes del sistema, roles jugados e identifican conversaciones entre los mismos; descomposición del sistema (agente) en subsistemas (componentes del agente) e interconexión de los mismos (definición de la arquitectura del agente mediante componentes); diagramas UML de despliegue para indicar cuántos agentes habrá en el sistema y de qué tipo.

Las comunicaciones entre diferentes elementos (componente-componente, agente-agente, rol-rol, tarea-tarea) se refieren al envío de estímulos desde una máquina de estados a otra.

[1] Ingeniería de software de sistemas multiagente.

En el caso de las tareas, a estas máquinas las denominan en MaSE *diagramas de tareas concurrentes*.

La herramienta de soporte, agentTool (DeLoach 2001)**,** permite generar código automáticamente a partir de la especificación del sistema, que en MaSe es el conjunto final de diagramas. La generación de código es independiente del lenguaje de programación utilizado, ya que se realiza recorriendo las estructuras de datos generadas en la especificación y generando texto como salida.

Dentro de la misma herramienta, MaSE incorpora utilidades para verificar la corrección de los protocolos que utilicen los agentes. MaSE usa intensivamente máquinas de estados para especificar el comportamiento de diversos elementos de la especificación. Aunque en MaSe no se menciona, esta idea no es original. Además de su aplicación en UML, se puede revisar la línea de investigación en *Abstract State Machines* (ASM) y el conocido SDL (Union 2002) que lleva utilizándose en industria desde hace dos décadas. En estas líneas existen multitud de herramientas, métodos de verificación, y desarrollos completos.

De todas formas, y aunque el diseño de sistemas distribuidos basándose en máquinas de estados ya ha sido aceptado industrialmente, no es tan sencillo verificar si con esta técnica se es capaz de expresar elementos característicos de la tecnología de agentes como el razonamiento de los agentes, organización de los agentes o caracterización de su estado mental. De hecho, estos son elementos que esta metodología no considera.

Respecto del proceso de desarrollo, MaSE propone un proceso más simple que el de MAS-CommonKADS. De acuerdo con los artículos publicados, la metodología podría traducirse como *tome la herramienta de soporte y rellene los diferentes apartados*. La ventaja de este enfoque es que se aprende rápido: sólo hay que experimentar con la herramienta. Sin embargo, la trampa de esta forma de modelar es que se obvia que, como en el modelo BDI, se tienen dependencias entre los diagramas propuestos (como entre los diagramas de secuencia y las conversaciones) y que no es tan sencillo el saber qué máquinas de estados definen la ejecución de una tarea en el contexto de una interacción.

En cuanto al proceso de generación de código, la versión actual integra el código de los componentes a generar con el código de la herramienta. Esto es, el código a generar aparece como cadenas de caracteres que se concatenan con información extraída de la

especificación. Esto quiere decir que cambios en el código de componentes generados implican la recopilación necesaria de la herramienta completa (Gómez 2003).

2.1.6 Metodología GAIA

GAIA (Wooldridge, Jennings et al. 2000) (Zambonelly, M et al. 2000) es una metodología para el diseño de sistemas basados en agentes cuyo objetivo es obtener un sistema que maximice alguna medida de calidad global (no se llega a detallar cuál). GAIA pretende ayudar al analista a ir sistemáticamente desde unos requisitos iniciales a un diseño que, según los autores, esté lo suficientemente detallado como para ser implementado directamente.

Esta metodología tiene el concepto básico de dos fases bien separadas, análisis y diseño (ver anexo 2b) (Zambonelly, M et al. 2000).

En particular en la fase de análisis se propone los siguientes pasos (Zambonelli et al 2000):

1. Identificar los roles en el sistema y definir una lista de roles importantes en un lenguaje de descripción informal.
2. Por cada rol, identificar los protocolos asociados, lo modelos de interacción que probablemente ocurran entre roles.
3. Elaborar los modelos de roles completos y modelos de interacciones y si es requerido iterar las etapas anteriores.

En la fase de diseño se proponen los siguientes pasos:

1. Identificar el modelo de agente, es decir, agregar roles en tipos de agentes y refinarlos.
2. Identificar los servicios que los agentes tienen que proveer en orden para que cumplan los roles asignados.
3. Desarrollar el modelo de conocimiento para identificar ineficiencias en el diseño y si es requerido iterar las etapas anteriores.

La metodología GAIA (Wooldridge, Jennings et al. 1998) (Wooldridge, Jennings et al. 2000) se centra en la idea de que la construcción de sistemas basados en agente es un proceso de diseño organizacional, es muy genérica (Julián and Botti 2003) sin referenciar ningún aspecto de implementación. Esto se consigue a través de la idea de *organización*. Una organización en GAIA es una colección de roles, los cuales mantienen ciertas

relaciones con otros y toman parte en patrones institucionalizados de interacción con otros roles. Los roles agrupan cuatro aspectos: responsabilidades del agente, los recursos que se le permite utilizar, las tareas asociadas e interacciones (Gómez 2003).

GAIA propone trabajar inicialmente con un análisis a alto nivel. En este análisis se usan dos modelos, *el modelo de roles* para identificar los roles clave en el sistema junto con sus propiedades definitorias y el *modelo de interacciones* que define las interacciones mediante una referencia a un modelo institucionalizado de intercambio de mensajes, como el FIPA-Request, este es un protocolo de interacción que permite a un agente pedir a otro agente que lleve a cabo una acción. El iniciador ejecuta la petición mediante el acto comunicativo request. El participante procesa el mensaje correspondiente y decide si aceptar o rechazar la petición. La respuesta tiene lugar mediante los actos comunicativos *agree* (opcional) y *refuse* respectivamente (González 2004).

Tras esta etapa, se entraría en lo que GAIA considera diseño a alto nivel. El objetivo de este diseño es generar tres modelos: el *modelo de agentes* que define los tipos de agente que existen, cuántas instancias de cada tipo y qué papeles juega cada agente, el m*odelo de servicios* que identifica los *servicios* (funciones del agente) asociados a cada rol, y un *Modelo de conocidos*, que define los enlaces de comunicaciones que existen entre los agentes.

A partir de aquí, los autores de GAIA proponen aplicar técnicas clásicas de diseño orientado a objetos. Sin embargo, GAIA declara que queda fuera de su ámbito. Esta metodología sólo busca especificar cómo una sociedad de agentes colabora para alcanzar los objetivos del sistema, y qué se requiere de cada uno para lograr esto último (Gómez 2003). La principal crítica que se puede hacer a GAIA es que se queda a un nivel de abstracción demasiado alto. Según los autores, con ello se consigue desacoplar GAIA de las distintas soluciones de implementación de agentes. Sin embargo, la utilidad de este desacoplamiento queda por demostrar. Dado el nivel de abstracción en que se queda es de esperar que el esfuerzo a invertir para pasar de una especificación GAIA hasta su implementación sea alto.

Un aspecto discutible es el uso combinado de fórmulas lógicas con fichas de documentación clásicas en ingeniería del software (Pressman 1982). En GAIA se resalta la

importancia del uso de fórmulas lógicas sin explicar los inconvenientes. Es cierto que se consigue mayor precisión, pero a costa de: invertir más trabajo, requerir desarrolladores más especializados que sepan comprenderlas, y definir la semántica de los predicados que se están empleando. Respecto del proceso de desarrollo, GAIA omite las distintas dependencias entre los modelos propuestos, lo cual es fundamental a la hora de proponer un proceso que dé como salida la especificación del sistema. Esta metodología se proyecta principalmente en los SMA como conjunto de entidades que interactúan (Romero 2010).

Para terminar, solo mencionar la falta de herramientas de soporte para esta metodología. De todas las revisadas es la que menos soporte tiene, lo cual sorprende dado el impacto que ha tenido (Gómez 2003).

2.1.7 Metodología INGENIAS

Esta metodología por su parte se basa en el trabajo de MESSAGE (Methodology for Engineering Systems of Software Agents) (EURESCOM 2000) (EURESCOM 2001) (Caire, Leal et al. 2002), la cual es una metodología orientada a agentes, que incorpora técnicas de ingeniería del software cubriendo el análisis y diseño de sistemas multiagente, esta provee un lenguaje, un método y unas guías de cómo aplicar la metodología (Julián and Botti 2003), ha tenido un gran impacto en la comunidad dedicada al estudio de los agente software. Como muestra de esta influencia, está la metodología INGENIAS (Gómez and Fuentes 2002), y otra que aplica las ideas de MESSAGE al desarrollo de sistemas en tiempo real, llamada RT-MESSAGE (Inglada 2002) (Inglada and Botti 2002). El motivo de presentar INGENIAS en lugar de MESSAGE, se debe a que se considera INGENIAS como evolución de las ideas de MESSAGE. INGENIAS profundiza en los elementos mostrados en el método de especificación, en el proceso de desarrollo, además de incorporar nuevas herramientas de soporte y ejemplos de desarrollo.

INGENIAS, como MESSAGE, define un conjunto de meta-modelos (una descripción de alto nivel de qué elementos tiene un modelo) con los que hay que describir el sistema.

INGENIAS reconoce cinco meta-modelos que describen al sistema (Gómez and Fuentes 2002) (Gómez and Pavón 2004), los cuales son nombrados modelos. Para describir un SMA por esta metodología se deben usar los siguientes modelos:

- Modelo de Agente. Describe agentes simples, sus tareas, metas, estado mental inicial y roles que juega. Los modelos de agente son usado para describir estados intermedios de agentes.
- Modelo Interacción. Describen como toma lugar la interacción entre agentes. Cada declaración de interacción incluye actores involucrados, objetivos perseguidos por la interacción y una descripción del protocolo que sigue la interacción.
- Modelo de objetivos y tareas. Describe relaciones entre objetivos y tareas, estructuras de objetivos y estructuras de tareas. Este es usado también para expresar cuales son las entradas y salidas de las tareas y cuáles son sus efectos en el entorno o en el estado mental de los agentes.
- Modelo Organización. Describe como los componentes del sistema (agentes, roles, recursos y aplicaciones) son agrupados, tareas que son ejecutadas en común, qué metas ellos comparten y que contraste existe en la interacción entre agentes.
- Modelo entorno. Define la percepción de agentes en términos de existencia de elementos del sistema. Este también identifica recursos del sistema y quien es responsable de administrarlos.

Los meta-modelos indican qué hace falta para describir un sistema: agentes aislados, organizaciones de agentes, el entorno, interacciones entre agentes o roles, tareas y objetivos. Estos meta-modelos se construyen mediante un lenguaje de meta-modelado como el GOPRR (*Graph, Object, Property, Relationship, and Role*) (Lyytinen and Rossi 1999). En la construcción de estos meta-modelos se integran resultados de investigaciones en forma de entidades y relaciones entre entidades. La instanciación de estos meta-modelos produce diagramas, los modelos, similares a los que se usa en UML[2], con la diferencia de que estos diagramas se han creado exclusivamente para definir el sistema multiagente.

El proceso de instanciación de los meta-modelos no es trivial. Existen muchas entidades y relaciones a identificar, además de dependencias entre distintos modelos. Por ello, INGENIAS (Gómez 2002) define un conjunto de actividades cuya ejecución termina en un

[2] Lenguaje unificado de modelado

conjunto de modelos. Estas actividades a su vez se organizan siguiendo un paradigma de ingeniería del software, el Proceso Unificado (Jacobson, Rumbaugh et al. 1999)

La ejecución de actividades para producir modelos se basa en la herramienta INGENIAS IDE (ver anexo 1c), una herramienta para modelado visual. Esta herramienta almacena la especificación del sistema utilizando XML (Lenguaje de marcas extensible) (González 2004) (Gómez 2003).

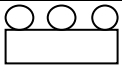
Desde la especificación en XML, se plantea el procesarla:

- Para generar código. Genera código rellenando plantillas de código con información de la especificación.
- Para generar la documentación. De una forma similar a la generación de código, se parte de plantillas de documentación que se completan utilizando información de los modelos.

INGENIAS se proyecta en Meta-modelos (Romero 2010), ello atenta en su contra a la hora de aplicaciones de desarrollos reducidos.

Como MAS-CommonKADS, INGENIAS dispone de una cantidad ingente de entidades y relaciones. Su uso mediante la herramienta de soporte INGENIAS IDE, se facilita pero se sigue requiriendo que el desarrollador revise la documentación para entender qué hace cada entidad y cuál es el propósito de cada relación.

El proceso de generación de código es más flexible que el ofrecido en ZEUS y MaSE. La principal diferencia consiste en que los desarrolladores pueden configurarlo a voluntad y adaptarlo a sus necesidades sin tener que modificar la herramienta de análisis/diseño. A continuación (Tabla 2.1) se muestran algunas notaciones con sus respectivas descripciones (Rodríguez 2009).

Notación	Descripción
	Representa una organización de agentes
	Representa un grupo de agentes
	Representa un rol que puede ser jugado por agentes

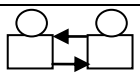
	Representa una interacción entre agentes
	Representa un agente
	Representa una tarea
	Representa un objetivo
	Representa un hecho
	Representa una aplicación interna al sistema multiagente
	Representa un evento
	Estado mental de un agente

Tabla 2.1 Algunas notaciones de INGENIAS.

Seguidamente se presenta de forma resumida una metodología de tiempo real.

2.1.8 Metodología RT-MESSAGE

Esta metodología tal y como se ha planteado está orientada a la construcción de Sistemas Multiagente de Tiempo Real, basándose en la metodología MESSAGE. Fundamentalmente se refiere a la idea de proveer los elementos necesarios para satisfacer requerimientos a partir de una especificación de un problema de tal forma que se pueda obtener directamente un prototipo ejecutable del sistema a desarrollar (Julián and Botti 2003). Se considera que los requerimientos no difieren respecto a otros paradigmas.

RT-MESSGE usa el mismo juego de modelos de MESSAGE (Julian and Botti 2004):

- Modelo de Organización (OM): Este modelo se enfoca principalmente en la estructura y la organización y del potencial de las suborganizaciones.
- Modelo de Tarea/Meta (GM): Este modelo se enfoca en la identificación de metas y tareas del sistema así como de sus relaciones.
- Modelo de agente (AM): En este modelo, todo agente o rol del sistema es especificado individualmente, indicando las principales características.

- Modelo de Interacción (IM): Este modelo se encarga de modelar las interacciones entre agentes o roles del sistema. La especificación puede definir protocolos de complejas interacciones.
- Modelo de dominio (DM): Este modelo especifica los conceptos y relaciones en el problema del dominio para ser resueltos.

La principal contribución de esta metodología en el modelo de organización son los refuerzos de los artefactos para la especificación de conducta del sistema, la posibilidad de usar diagramas con marcas temporales en la especificación de conducta.

Los sistemas a desarrollar se basan en la plataforma de sistemas multiagente de tiempo real SIMBA (Soler, Julián et al. 2002) (Julián, Carrascosa et al. 2002), la cual permite diseñar las interacciones, organización y estructura interna de los agentes de tiempo real del sistema. El principal componente de SIMBA es la arquitectura de agente de tiempo real ARTIS (Botti, Julián et al. 1999). Por medio de ARTIS se dispone de una arquitectura de agentes donde confluyen características propias de sistemas de tiempo real estricto con componentes inteligentes. En el método propuesto únicamente se consideran las actividades de análisis, diseño e implementación. De manera resumida en el análisis del sistema se trata de especificar de la forma más apropiada posible el problema a resolver, para ello se emplean los modelos propuestos por MESSAGE ampliados para poder modelar las características propias de sistemas de tiempo real. Por lo que respecta al diseño, toma como entrada todos los artefactos generados en el análisis, centrándose en la transformación de las entidades especificadas en dichos artefactos en entidades computacionales, para ello se emplea también la arquitectura de sistema multiagente de tiempo real SIMBA. Finalmente se propone la implementación del diseño realizado, es decir, de los componentes, interacciones y demás elementos diseñados. Para ello, se dispone de una herramienta de desarrollo que permite la obtención de un prototipo directamente ejecutable del sistema. (Julián and Botti 2003).

2.1.9 Metodología PROHA.

Otra metodología de interés a tratar en este trabajo es **PROHA** (Product, Resource, Order Heterarchical Architecture) (Cenjor and García 2005).

Es un modelo construido a partir de PROSA (Van Brussel, Wyns et al. 1998), del cual se toma como referencia el proceso de agentificación de los productos, recursos y órdenes del sistema. El proceso de diseño se basa en el planteamiento de un sistema heterárquico de agentes deliberativos, en concreto, de agentes BDI. Esta metodología se adapta a los sistemas de grandes dimensiones y alta complejidad ya que permite crear sistemas software que representan cada uno de los elementos que configuran el sistema productivo a fin de facilitar su gestión y control. Estos agentes interactúan de forma heterárquica a pesar de estar en sistemas multinivel. Esta circunstancia hace que los procesos de toma de decisiones sean más efectivos que en otras metodologías, ya que los procesos de coordinación y negociación de iniciativas pueden ser llevados a cabo por cualquier tipo de agente, sin tener en cuenta su peso específico real. La metodología de PROHA se centra principalmente en tres procesos (Cenjor and García 2005), (i) la agentificación o identificación de agentes, (ii) la definición de los agentes y de las características de diseño y (iii) la definición de los protocolos de interacción.

El *proceso de agentificación* (Cenjor and García 2005), responde a un proceso estructurado en cuatro pasos. En primer lugar, se realiza la *identificación y determinación de las clases.* Este proceso requiere de un alto grado de abstracción para ser eficaz. Consiste en observar el sistema productivo de forma global y en pleno funcionamiento, reconocer los distintos elementos que lo forman e intentar agrupar los elementos en conjuntos funcionales básicos, que serán las clases.

En segundo término se procede a la *identificación de los agentes que integran cada clase* mediante la asociación a los elementos físicos que lo integran. Así pues, hay que atender a los elementos que, por sus funciones u objetivos, puedan integrarse dentro de las clases ya determinadas. Una vez identificados, se les asocia al sistema como agentes.

El siguiente paso es la *creación de la taxonomía de agentes*, *consiste* en catalogar los agentes según sea su condición en tres grupos bien diferenciados: productos, recursos y órdenes Los productos son todo aquel fragmento de material que se encuentre dentro del sistema (o en espera) y que sea susceptible de sufrir alguna operación de transformación. Se considera recurso a todo lo que pueda asistir tanto a productos como a otros recursos. Toma carácter de orden todo aquel elemento que se integre en la burocracia del sistema y que

determine cuales son las condiciones y los pormenores con los que debe trabajar el sistema. Por último, la *definición de la estructura*, todos los agentes dentro del sistema se encontrarán en un mismo nivel, de forma que todos pueden interactuar con todos, por tanto, se trata de una heterarquía.

Por su parte la *definición de los agentes* es la *fase de diseño* (Cenjor and García 2005) donde se establecen la estructura de los agentes y sus características de diseño. Por tanto, la estructura del diseño de cada uno de estos agentes se divide en cuatro partes: tres de ellas dedicadas a la definición de los deseos, creencias e intenciones, y la última dedicada al análisis de las percepciones obtenidas del entorno por medio de los sensores.

Las *creencias* determinan la base de conocimientos del agente, es la información que debe tener fija cada agente para poder realizar la toma de iniciativas, mediante procesos de negociación y coordinación. Los *deseos* definen cuales son las pretensiones que tienen los agentes dentro del sistema productivo aún cuando no pueda conseguir siempre satisfacerlas. Estos deseos determinan las posibilidades de actuación, mediante las creencias y las intenciones. Las *intenciones* representan aquellos estados del sistema a los que los agentes se comprometen a conseguir. Las intenciones se actualizan constantemente gracias a los deseos, creencias y las propias intenciones en un instante anterior, con el fin de determinar hasta qué punto pueden ser ejecutadas. (Montalvo et al. 2005)

Los protocolos de interacción PROHA, se basan en la coordinación, en PROHA esta propiedad está implícita en los agentes. En el establecimiento de la coordinación, la dificultad viene, en primer lugar, porque las acciones de los agentes se pueden interferir unas con otras; en segundo lugar, porque puede haber restricciones globales que acoten el diseño; y finalmente, porque un solo agente no es capaz de alcanzar los objetivos del sistema sino que necesita de otros agentes (Bussmann, Jennings et al. 2004) La estructura de agentes es como un mercado de competencia perfecta (Cenjor and García 2005). Al igual que en la vida real, es preciso, para obtener los mayores beneficios, disponer de información veraz y reciente. Esta necesidad queda cubierta ya que la tecnología RFID-IMS asume las funciones de identificación de objetos y de la gestión de la información.

- *Tecnología RFID-IMS*[3] (Radio Frequency Identification – Information Management System)

Se trata de una tecnología que fue concebida con el objetivo de soportar todo el tratamiento de la información que se genera en los sistemas de gran dinamismo y proporcionar dicha información al sistema para poder realizar la toma de decisiones de forma instantánea. Es decir, es un sistema que permite flexibilizar el acceso a la información.

RFID es la tecnología de identificación por radiofrecuencia. Se basa en tarjetas de radiofrecuencia y antenas o sensores RF de lectura y/o escritura (según se precise). Los sensores RF son aparatos capaces de crear campos electromagnéticos a diferentes frecuencias y en distintos áreas en función de su potencia. Son los encargados de obtener la información incluida en las tarjetas y de escribir en ellas los datos que procedan en cada momento.

IMS es un sistema de gestión de la información; en este caso, de la información que almacenan las tarjetas. Toda la información se gestiona mediante bases de datos. Se puede gestionar la información mediante un computador y un servidor que aloje la base de datos.

Se puede globalizar que estos sistemas RFID-IMS aplicados a SMA suponen una producción inteligente y flexible. El MAS controla el sistema de forma instantánea. El ser humano solo tiene que determinar las características del diseño y la implementación (Cenjor and García 2005).

En el anexo 3 se muestra una comparación de algunos de los aspectos abordados anteriormente en el capítulo para cada una las metodologías estudiadas..

Si se desea seleccionar la mejor metodología, ¿cuál sería la ganadora? Esta pregunta no tiene una respuesta fija. Cada metodología, por el bagaje de sus creadores, se especializa en áreas concretas. Si se está acostumbrado a trabajar con sistemas basados en conocimiento, lo lógico es que se elija MAS-CommonKADS. Si la experiencia del usuario está en el área de los objetos, la recomendación sería de MaSE. Si por el contrario está interesado en un enfoque más orientado a agentes, puede seleccionar ZEUS, INGENIAS, BDI o GAIA. Y si

[3] Identificación por radiofrecuencia (RFID)-sistema de gestión de la información (IMS).

lo que se quiere es tener un soporte de herramientas, la lista de metodologías válidas se reduce considerablemente: ZEUS, MaSE, INGENIAS. En los casos en que se requiera un proceso de desarrollo robusto, detallado y ensayado en desarrollos reales, la recomendación sería MAS-CommonKADS o INGENIAS. Si solo interesa una visión superficial del sistema, sin entrar en detalles de diseño e implementación, se recomendaría GAIA. Si se busca una metodología desarrollada a partir de modelos formales, entonces sería BDI la mejor elección. Otra opción que tienen los desarrolladores es quedarse con lo que más les interese de cada metodología, una opción sería usar la técnica de modelado de INGENIA junto con su herramienta de especificación e implementar con ZEUS. Una metodología que brinda mejoras notables para sistemas de grandes dimensiones y alta complejidad es PROHA, logrando un paso de avance con la inclusión de la tecnología RFID-IMS.

Resumiendo se puede afirmar que no existe una metodología mejor ni peor más bien se trata del área de aplicación que desee el usuario.

Para la propuesta de SMA que se realizará posteriormente se requiere un enfoque orientado a agentes, un proceso de desarrollo robusto, detallado y ensayado en desarrollos reales y un buen soporte de herramientas.

Tales requerimientos quedan cubiertos por la metodología INGENIAS.

Esta metodología se basa en MESSAGE por lo tanto está orientada a agentes, que incorpora técnicas de ingeniería del software cubriendo el análisis y diseño de sistemas multiagente. Para cubrir la necesidad de un proceso de desarrollo robusto, detallado y ensayado en desarrollos reales INGENIAS reconoce cinco meta-modelos que describen el sistema, a estos también se les denomina modelos, estos indican qué hace falta para describir un sistema: agentes aislados, organizaciones de agentes, el entorno, interacciones entre agentes o roles, tareas y objetivos. Estos meta-modelos se construyen mediante un lenguaje de meta-modelado como el GOPRR, constituyendo una gran ventaja para el diseñador.

Dicha metodología posee una buena herramienta, INGENIAS IDE (anexo 1c), esta permite la ejecución de actividades para producir modelos, es una herramienta para modelado visual. Esta herramienta almacena la especificación del sistema utilizando XML,

permitiendo la generación de código y de documentación que indudablemente facilita el desarrollo del SMA.

Una mejora que se introduce es la incorporación a la metodología la posibilidad del trabajo en equipo. Si bien una persona puede completar la especificación de todo el sistema, cuando varias personas intentan completar diferentes aspectos del sistema, surgen problemas. Los modelos presentados son compatibles con desarrollos en grupo siempre y cuando se establezcan parcelas de actuación centradas en flujos de trabajo donde los actores sean roles, lo cual constituye una ventaja para el desarrollo de un SMA.

Ante tales argumentos INGENIAS se convierte en la metodología seleccionada para la propuesta del SMA.

Conclusiones parciales

Al finalizar este capítulo se confirma el hecho de que las metodologías abordadas constituyen una referencia obligatoria para el trabajo con SMA.

No existe la mejor metodología, el asunto está en analizar el área de aplicación que se requiera.

Evidentemente para desarrollar la implementación SMA en aplicaciones industriales o de la rama de control de procesos se necesita acudir a una metodología que brinde un buen soporte de herramientas reforzado con una sólida base conceptual capaz de reflejar las características que rigen el sistema.

Se evidencia que a pesar de las facilidades que brindan las metodologías el proceso de desarrollo de los SMA requiere un alto grado de abstracción para poder abarcar una serie de aspectos que son imprescindibles para la concepción de un SMA.

Dadas las condiciones establecidas para desarrollar la propuesta de SMA, la metodología que sobresale es INGENIAS.

CAPÍTULO 3. PROPUESTA DE UN SISTEMA MULTIAGENTE PARA LA IDENTIFICACIÓN DE UNA PLANTA DE TRATAMIENTO DE RESIDUALES.

En el presente capítulo se propone un SMA para la identificación de sistemas en procesos biotecnológicos de tratamientos de residuales, este sistema presenta la ventaja de que una vez desarrollado, se puede extender su aplicación a otros procesos que intervienen en el tratamiento de residuales.

Antes de presentar la estructura de organización del SMA propuesto, se debe comprobar el cumplimiento de una serie de requerimientos para el sistema multiagente expresados en los siguientes elementos:

1. Un entorno, este expresa o describe el medio en que se desenvuelve el agente y el SMA en general.

2. Un conjunto de objetos. Estos objetos se encuentran integrados con el entorno, es posible en un momento dado asociar uno de estos objetos con un lugar en el entorno. Estos objetos son pasivos, pueden ser percibidos, creados, destruidos y modificados por agentes.

3. Un conjunto de agentes, que representan las entidades activas del sistema, deben realizar una serie de operaciones, que hacen posible que los agentes perciban, produzcan, consuman, transformen y manipulen objetos.

4. Un conjunto de relaciones que unen objetos, y, por lo tanto, agentes.

5. Operadores que representan la aplicación de operaciones sobre el mundo y la reacción de éste al ser alterado. Estos operadores se pueden entender como *las leyes del universo*.

Como se comentó anteriormente, la propuesta de este SMA está elaborada siguiendo la metodología INGENIAS.

Dado que la instanciación de los modelos propuestos por INGENIAS no es trivial evidenciándose en los diagramas y modelos que se producen, normalmente se utiliza la herramienta INGENIAS IDE que posibilita la ejecución de actividades que facilitan la solución a tal inconveniente, ayudando así a la implementación del SMA, sin embargo en el momento de desarrollo del presente trabajo no se ha adquirido dicha herramienta, y el tiempo que se requiere para la implementación del sistema propuesto es mayor que el período con que se cuenta para la discusión del presente proyecto. Ante tales limitaciones se ha decidido presentar la descripción de los meta-modelos (modelos) teniendo en cuenta las principales características que debe cumplir el SMA propuesto, de tal forma que sirvan de base para su implementación. Se usan las notaciones de INGENIAS.

Se deja abierto para posteriores trabajos la descripción del modelo de entorno y la implementación de estos modelos.

Antes de describir los modelos se debe adicionar información sobre algunos términos como son *estado mental, rol, aplicaciones, eventos y hechos.*

- Estado mental: Estará compuesto por *entidades mentales* que tendrán que contemplar como mínimo creencias, compromisos y deseos. Se admiten múltiples instancias del estado mental relacionadas con un agente (Gómez 2002).
- Rol: Se maneja como en MESSAGE, donde es empleado para modelar diferentes comportamientos que posteriormente un agente dado podría llevar acabo (Julián and Botti 2003).
- Aplicaciones: Por medio de las cuales se pueden habilitar tareas. Pueden emplearse para modelar servicios pasivos, esto es, un conjunto de operaciones que no requiere la interacción con ningún agente (Gómez 2002).
- Eventos: Cambios ocurridos, en el mundo que el agente capta (Gómez 2002).
- Hechos: Información cierta para el agente (Gómez 2002).

3.1 Modelo de organización de agentes.

En la figura 3.1 se muestra el primero de los modelos de la propuesta de SMA que es el modelo de organización de agentes, en el cual se muestra de manera general la estructura del SMA propuesto.

La organización de agentes se divide en dos grupos lógicos de agentes: uno relacionado con la identificación paramétrica *IdentParamétrica* y otro con la identificación no paramétrica *IdentNoParamétrica*, solamente se ha desarrollado la primera de estas pues la *IdentNoParamétrica* no es de interés para la aplicación de esta investigación.

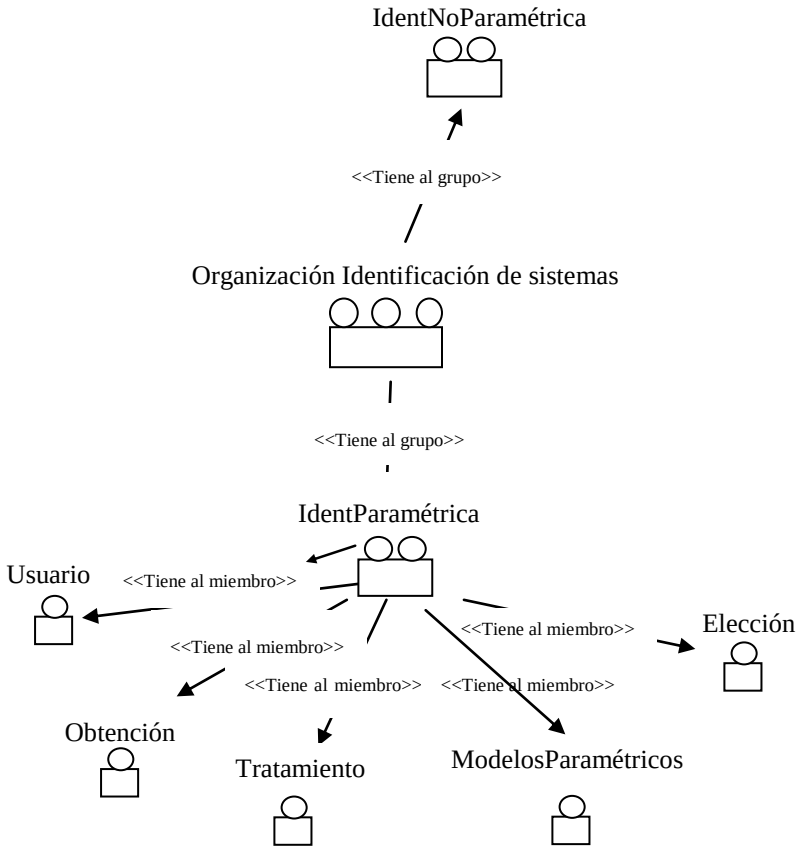


Figura 3.1 Modelo de organización de agentes.

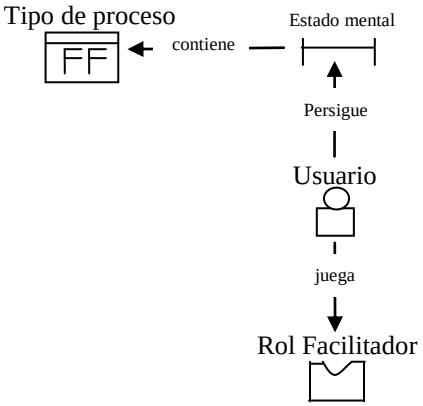
IndentParamétrica: agrupa a los agentes que tienen que ver con la identificación paramétrica del sistema que se esté analizando. Los agentes *Usuario*, *Obtención, Tratamiento, ModelosParamétricos y Elección* juegan en el SMA distintos roles los cuales son realizados mediante tareas que en varios casos provocan interacciones entre los agentes

que contribuyen al cumplimiento de los objetivos individuales de cada agente para lograr el objetivo general del grupo.

3.2 Modelos de agentes y roles, tareas y objetivos.

La figura 3.2 muestra el modelo el agente *Usuario.* Su estado mental contiene el tipo de proceso y juega un rol de Facilitador.

El rol que este agente juega (Fig. 3.3) lo hace apoyándose en dos tareas, *Especificar tipo de proceso* y *Depositar datos*.



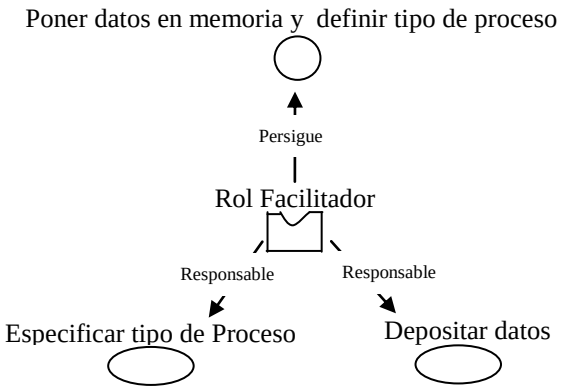
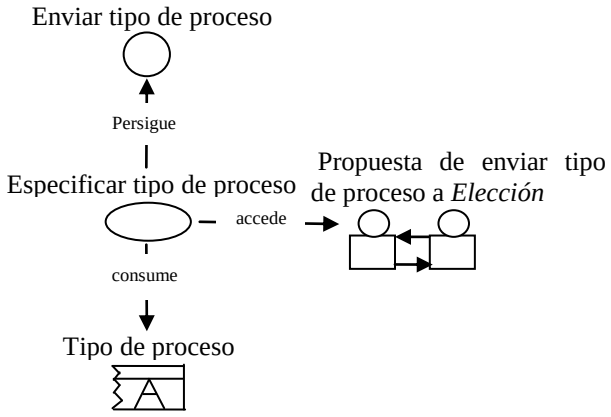


Figura 3.2 Modelo del agente: Usuario. Figura 3.3 Modelo del rol Facilitador.

La tarea *Especificar tipo de proceso* (Fig. 3.4) persigue el objetivo de enviar tipo de proceso para ello se utiliza el hecho tipo de proceso, este debe ser introducido por un usuario, y por medio de una interacción le envía el tipo de proceso al agente *Elección.*



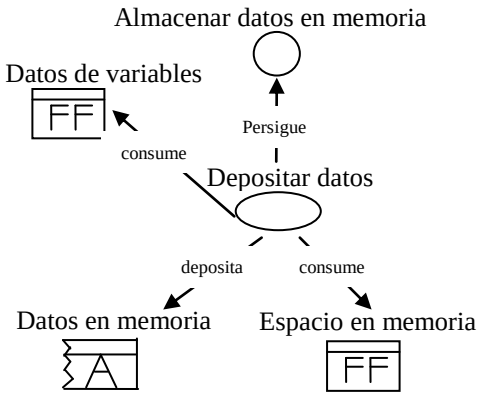


Figura 3.4 Modelo de la tarea Especificar tipo de proceso. Figura 3.5 Modelo de la tarea Depositar datos.

La tarea *Depositar datos* (Fig. 3.5) posee el objetivo de almacenar datos en memoria consume los datos de variables y espacio en memoria para depositar dichos datos en memoria.

La figura 3.6 muestra el modelo el agente *Obtención.* El estado mental de este agente, contiene un hecho que almacena los valores de las variables del proceso. Dicho agente juega en el sistema el rol *Interfaz* (fig. 3.7) que se describe a continuación.

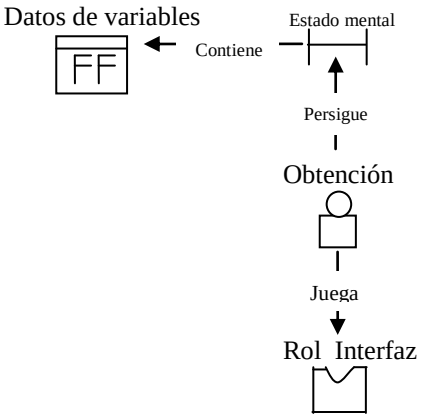


Figura 3.6 Modelo del agente: Obtención.

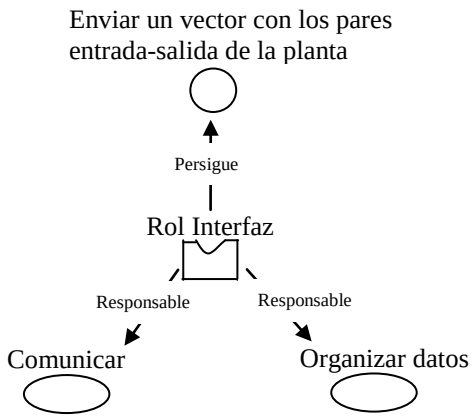


Figura 3.7 Modelo del Rol Interfaz.

Las capacidades del rol *Interfaz* se expresan como dos tareas, estas son, *Comunicar* (se encarga de comunicar con SMA, fig. 3.8) y *Organizar datos* (fig. 3.9).

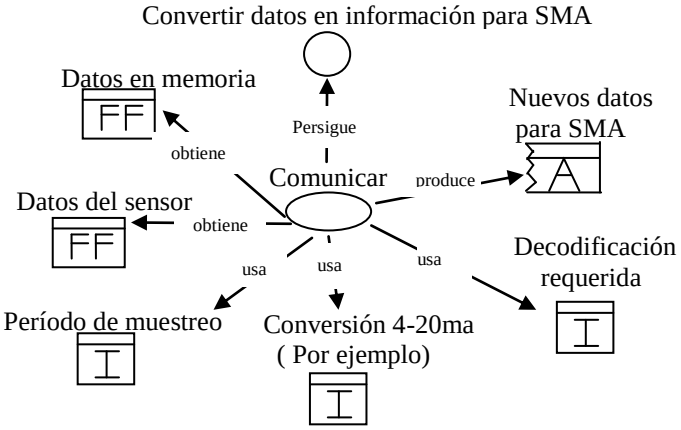


Figura 3.8 Modelo de la tarea: Comunicar con SMA.

La tarea *Comunicar* tiene como objetivo convertir datos en información compatible para el SMA, se encarga de cada un período de muestreo obtener los datos provenientes de un sensor o datos en memoria y mediante conversión 4-20 ma por citar un ejemplo o alguna decodificación requerida producir nuevos datos (entendibles) para SMA.

La siguiente figura muestra una serie de hechos, eventos e interacciones que son vitales para la satisfactoria ejecución de la tarea *Organizar datos* (Fig. 3.9).

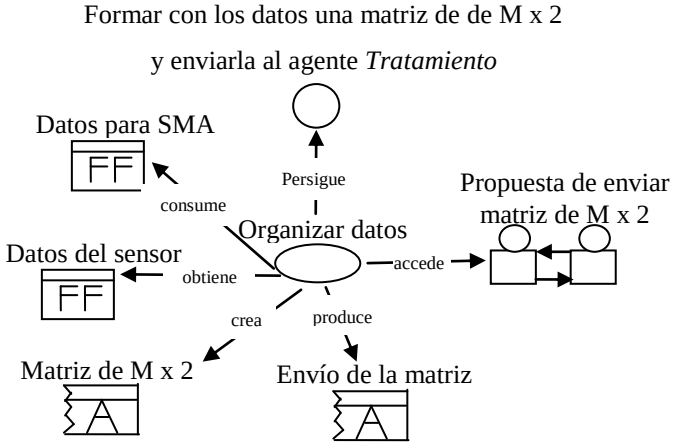


Figura 3.10 Modelo de la tarea: Organizar datos.

La tarea *Organizar datos* persigue formar con los datos una matriz de M (filas) x 2 columnas para enviarla al agente *Tratamiento*, esta tarea con los datos para SMA crea una matriz de M x 2 y mediante la interacción con el agente *Tratamiento*, si este acepta se produce el envío de dicha matriz.

Seguidamente se analizará el modelo del agente *Tratamiento* (fig. 3.11), el estado mental de este contiene un hecho que es el encargado de almacenar los datos de la matriz de M x 2. Este juega dos roles en el SMA: *Procesador de dato* y *Almacén.*

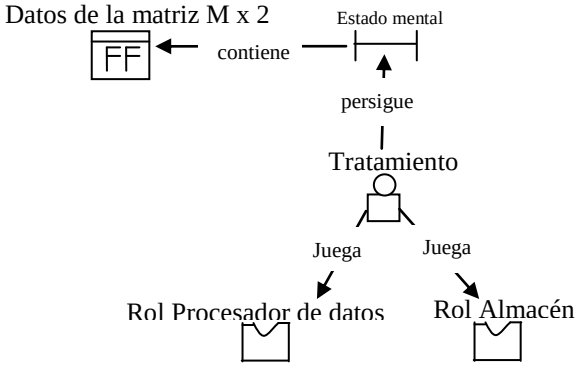
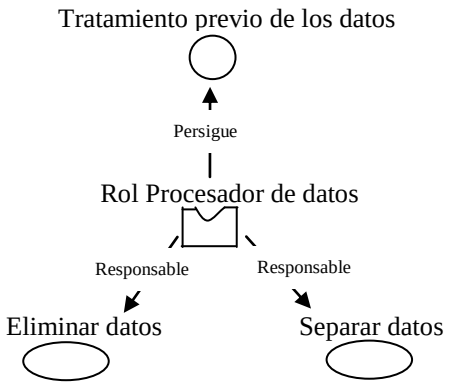


Figura 3.11 Modelo del agente: Tratamiento

El *Rol Procesador de datos* (Fig. 3.12a) tiene como objetivo el tratamiento previo de los datos, así posteriormente se usarán solamente los que aporten información al modelo, se logra con la realización de dos tareas, *Separar datos* y *Eliminar datos*.

El rol *almacén* persigue el objetivo de almacenar datos de identificación y validación y su mejor modelo paramétrico, para ello se ejecuta la tarea G*uardar en base de datos* (Fig. 3.12b).



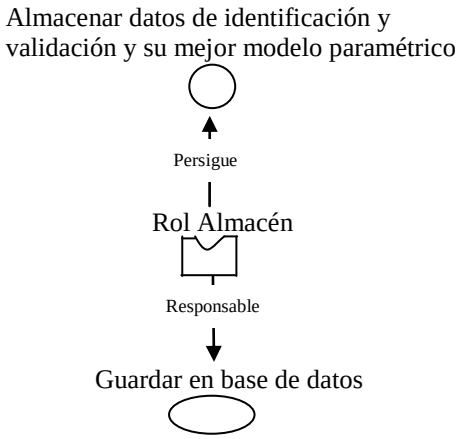


Figura 3.12aModelo del Rol Procesador de datos. Figura 3.12b Modelo del Rol Almacén.

La tarea *Eliminar datos* (Fig. 3.13) se propone quitar los datos que no aporten al modelo, para cumplirlo, a la matriz de M x 2 le aplica filtros, le elimina valores medios y valores fuera de rango, esta tarea posteriormente produce la matriz de M x 2 tratada, es decir, solamente contiene datos que realmente aportan información al modelo que se busca.

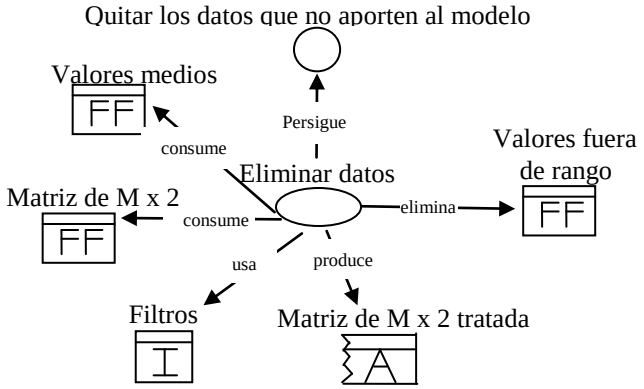


Figura 3.13 Modelo de la tarea: Eliminar datos.

La tarea *Separar datos* (Fig. 3.15) tiene como objetivo seleccionar una parte de los datos para identificación y la otra para validación, usa la matriz de M x 2 tratada y produce una mitad de los datos para identificación y la otra para validación, posteriormente envía los datos de identificación y validación por separados al agente *modelo paramétrico* cuando este acepte la propuesta.

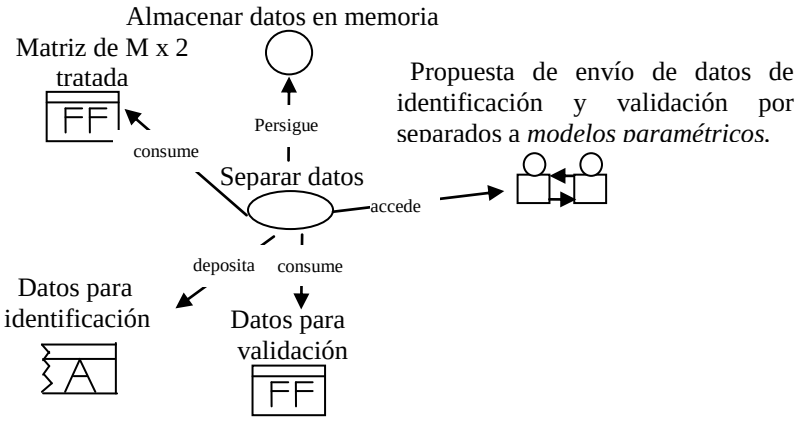


Figura 3.14 Modelo de la tarea: Separar datos.

Retomando el *Rol Almacén*, se apoya en la tarea *Guardar en base de datos* (Fig. 3.15). Esta tarea continúa con el mismo objetivo que el presente rol, para ello consume la reserva de espacio en memoria, produce el almacenamiento de estos valores con su mejor modelo paramétrico en la base de datos por tanto esta tarea también modifica la base de datos.

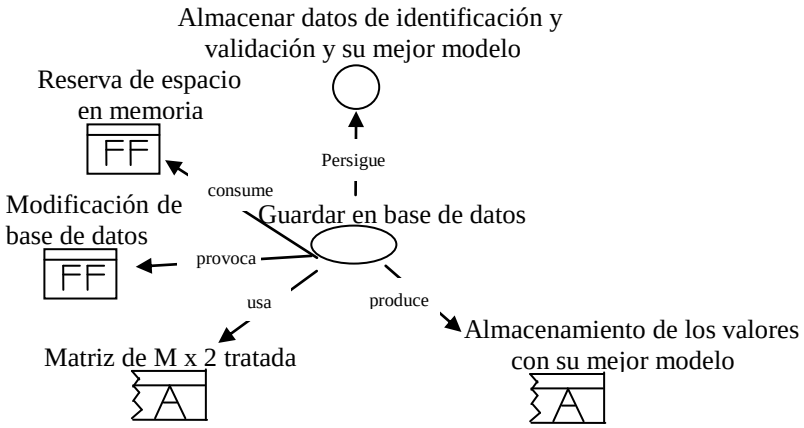


Figura 3.15 Modelo de la tarea: Guardar en base de datos.

A continuación se analiza el agente *modelo paramétrico* (Fig. 3.16) el cual contiene los datos de la matriz M x 2 tratada, este juega en el sistema el rol Proveedor modelo.

Dicho rol (Fig. 3.17) persigue como objetivo implementar tres modelos paramétricos, este expresa sus capacidades en las tres tareas que se le han asignado, TARX, TARMAX y TOE.

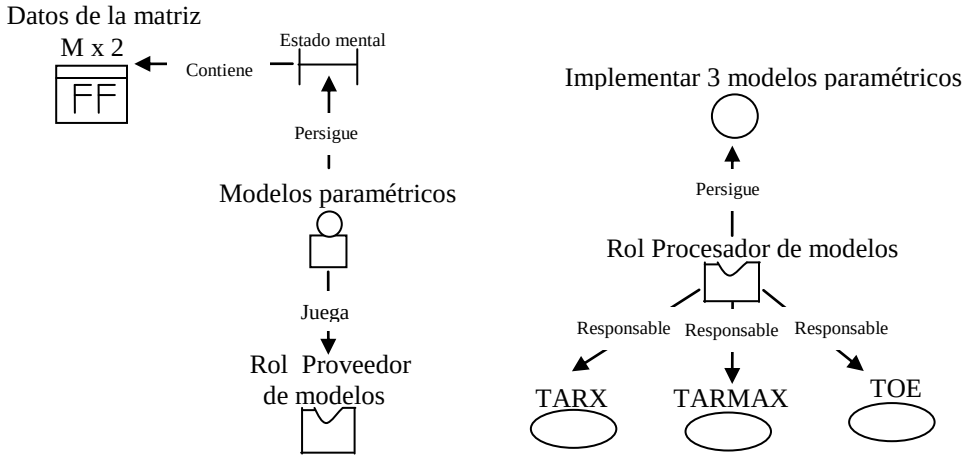


Figura 3.16 Modelo del agente: Modelos paramétricos. Figura 3.17 Modelo del Rol Proveedor de modelos.

La tarea ARX (Fig. 3.18) tiene el objetivo de implementar el modelo ARX, usa la matriz M x 2 tratada y utiliza la aplicación modelo ARX, posteriormente se nutre de los criterios de selección para escoger la mejor estructura ARX, estos criterios aún cuando contengan aspectos de la validación como el fit y la varianza entre otros, van más allá del aspecto teórico, pues también escogen la mejor estructura de acuerdo a aspectos que analizaría un usuario, como son el número de coeficientes, y otros que tienen en cuenta el grado de dificultad a la hora de la implementación en la práctica de aplicaciones donde se necesite este modelo.

Después de seleccionar la mejor estructura ARX mediante una interacción se le propone al agente *elección* el envío de dicha estructura con sus coeficientes, si este acepta se le envía y así se hace también con los modelos ARMAX y OE.

Este mismo análisis se cumple para las dos tareas restantes, por tanto, en aras de no hacer muy extenso el trabajo, se decidió no incluir en el documento las restantes 2 tareas.

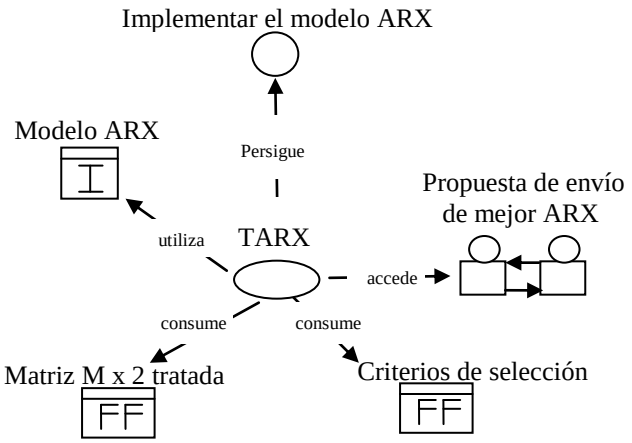
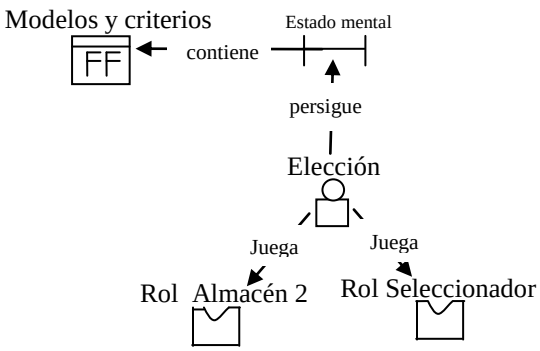


Figura 3.18 Modelo de la tarea: TARX.

Como último agente a analizar en esta propuesta de SMA se tiene a *Elección* (Fig. 3.19), con un estado mental que contiene modelos y criterios, estos facilitan la selección del mejor modelo; juega en el SMA el rol de *Seleccionador* y de *Almacén 2.*



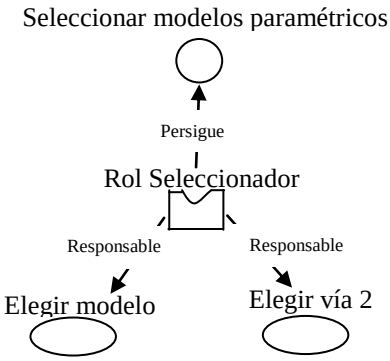


Figura 3.19 Modelo del agente: Elección.

Figura 3.20 Modelo del Rol Seleccionador.

El rol *Seleccionador* (Fig. 3.20) se traza como objetivo seleccionar el mejor modelo paramétrico y realiza para ello dos tareas, *Elegir mejor modelo* y *Elegir por vía 2.*

La tarea *Elegir mejor modelo* (Fig. 3.21) persigue el objetivo de seleccionar el mejor modelo implementado, para ello utiliza la mejor estructura del modelo ARX, la mejor del ARMAX y la mejor del OE y pasa a analizar los criterios de selección, los cuales están muy vinculados a los que se explicaron anteriormente, y posteriormente indica el mejor modelo, lo cual era el objetivo central de este trabajo, su manipulación y adquisición por otros agentes para las aplicaciones de control por ejemplo, será objetivo de futuras investigaciones.

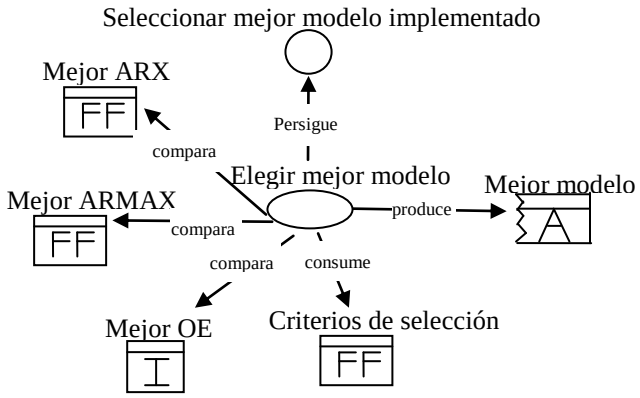


Figura 3.21 Modelo de la tarea: Elegir mejor modelo

La tarea *elegir por vía 2* (Fig. 3.22) tiene como objetivo seleccionar directamente el mejor modelo paramétrico , esta consume los hechos <<datos estadísticos para seleccionar mejor modelo para proceso en análisis>> y <<Tipo de proceso>>, en caso que no existan datos estadísticos para seleccionar mejor modelo para el proceso en análisis a partir del tipo de proceso especificado por el agente *Usuario,* el agente *Elección* envía la propuesta de implementar los tres modelos al agente *Modelos paramétricos* a partir de los datos que este último agente ha recibido del agente *Tratamiento.*

Si existen datos estadísticos para seleccionar directamente el mejor modelo para el proceso especificado por el agente *Usuario,* mediante la interacción con *modelos paramétricos* se solicita la implementación del mejor modelo para ese tipo de proceso. Posteriormente el agente *Elección* recibe el modelo implementado, si cumple con los criterios de validación entonces se califica como mejor modelo.

En caso de no cumplir, manda la propuesta al agente *Modelos paramétricos* de que implemente los restantes dos modelos para después mediante la tarea *Elegir mejor modelo* seleccionar el mejor.

El rol *Almacén 2* (Fig. 3.23) tiene el objetivo de almacenar mejor modelo en una base de datos, la tarea que lo acompaña es *Guardar mejor modelo* (Fig. 3.24).

Esta tarea continúa con el mismo objetivo, toma el mejor modelo para producir su almacenamiento en la base de datos, este evento provoca la modificación de la base de datos.

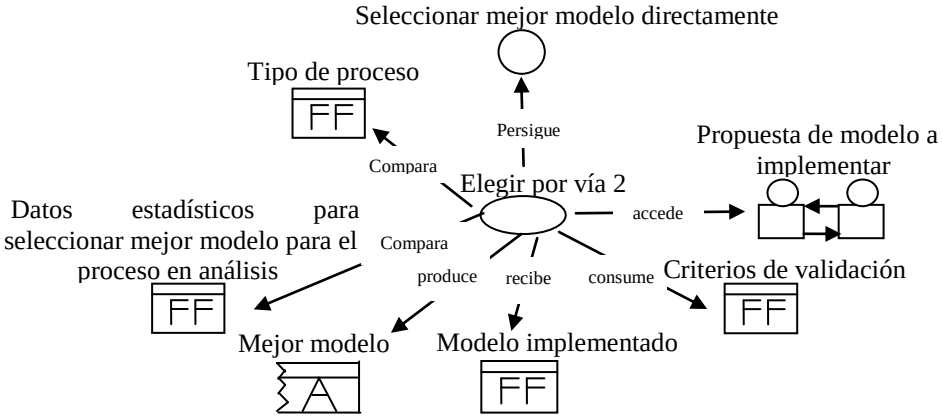
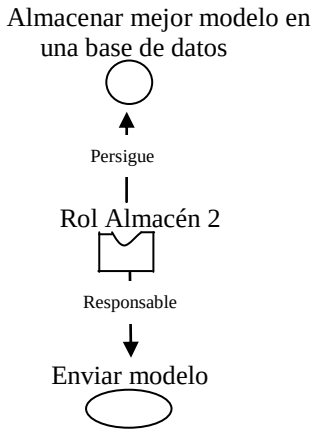


Figura 3.22 Modelo de la tarea: Elegir por vía 2.



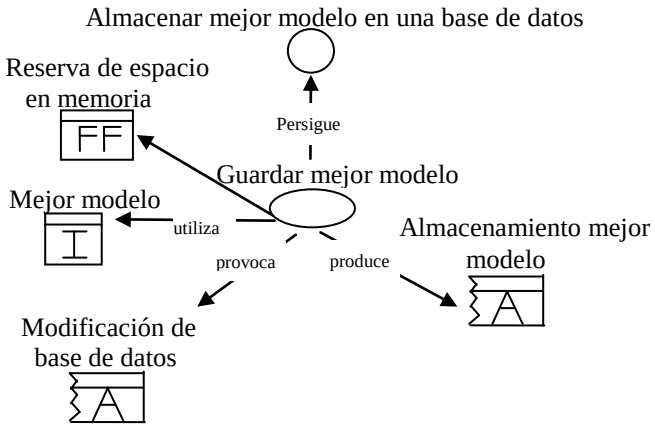


Figura 3.23 Modelo del rol Almacén 2. Figura 3.24 Modelo de la tarea: Enviar modelo

3.3 Modelo de interacciones.

Las interacciones identifican dependencias entre los componentes y contribuyen a la especificación del comportamiento de los componentes así como la funcionalidad asociada. Estos aspectos cubren un alto porcentaje de los asuntos a considerar cuando se diseña un sistema.

Se han identificado un total de 7 interacciones en esta propuesta.

- Interacción entre agente *Obtención* y *Tratamiento.*
- Interacción entre agente *Tratamiento* y *Elección.*

- Interacción entre agente *Tratamiento* y *Modelos paramétricos.*
- Cuatro interacciones entre *Modelos paramétricos y Elección.*

Todas con actos de habla: *proposal*

La interacción consta de dos roles, el «Rol-Colaborador» y el «Rol-Receptor». Siendo el «Rol-Colaborador» el agente que colabora proponiendo, soluciones, datos, decisiones y siendo « Rol-Supervisor» el agente que decide si aceptar la propuesta o no.
La interacción está basada en el protocolo «Propose» del estándar FIPA (González 2004), donde el acto del habla propose se utiliza para indicar que ejecutará una acción si el participante acepta, este responde a través de los actos comunicativos acept-proposal y reject-proposal para comunicar la decisión de aceptación o rechazo de la propuesta respectivamente.

A continuación se modela la primera de ellas (Fig. 3.28), las siguientes no se modelarán pues tienen el mismo acto de habla y desempeñan los mismos roles que se abordarán en este ejemplo.

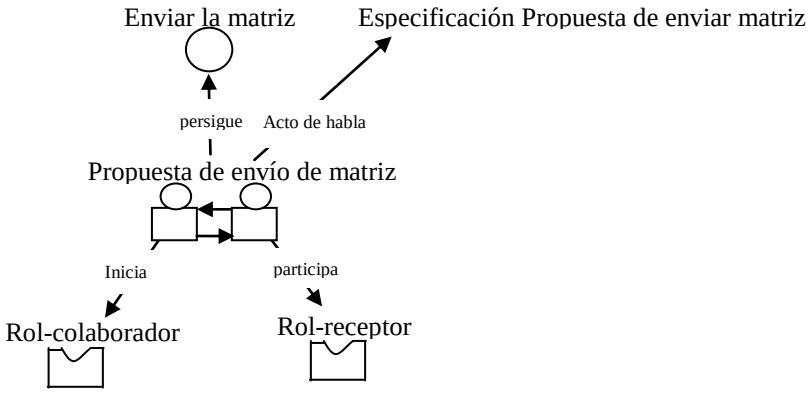


Figura 3.25 Modelo de interacción de propuesta de envío de matriz.

Conclusiones parciales

El proceso de desarrollo de un SMA presenta un elevado grado de complejidad, aspecto que se entiende a partir de las disímiles tareas y aplicaciones que puede realizar en la línea de trabajo que este abarca, de tal forma que simula el comportamiento humano en situaciones tan complicadas como a la hora de tomar decisiones.

La metodología seleccionada permite amplias posibilidades de desarrollo, evidenciadas en esta propuesta, a pesar del grado de abstracción, propiedad en mayor o menor medida para todas las metodologías, pero, de forma general es un atributo inherente para la descripción de los SMA.

La descripción del sistema mediante los meta-modelos permite abarcar satisfactoriamente los casos, aplicaciones y aspectos que rigen o influyen en el SMA, como son los roles, tareas e interacciones que protagonizan los agentes.

Mediante la propuesta de SMA se demostró que INGENIAS posee las condiciones para imponerse como unas de las metodologías referencia para el desarrollo de SMA, sobre todo en la parte de las aplicaciones ingenieriles o industriales tanto en el área del comercio como en el control de procesos.

CONCLUSIONES y RECOMENDACIONES

Al finalizar el presente trabajo se han podido obtener varios elementos importantes dignos de considerar en este apartado, por lo que se puede concluir que:

Los SMA son una respuesta eficaz para la identificación de sistemas en procesos complejos como los biotecnológicos y con especificaciones cada vez más exigente hasta el nivel de exigir cierto grado de autonomía de algunos controladores.

Se evidenció el alto grado de abstracción para el trabajo con agentes inteligentes y específicamente para desarrollar el SMA propuesto.

Se ha comprobado que el estudio de las metodologías existentes para usar en SMA constituye una referencia obligatoria si se quiere organizar el trabajo y abarcar un mayor espectro de los detalles y características necesarias para un buen desempeño del SMA.

Debido a la necesidad de una metodología que posea una orientación a la teoría de agentes e ingeniería del software, una robusta herramienta de desarrollo, además de contar con una buena documentación, es que se selecciona la metodología INGENIAS para el desarrollo de la propuesta, además de que ha tenido gran impacto en el desarrollo de SMA.

RECOMENDACIONES

Con el objetivo de desarrollar esta línea de trabajo se recomienda:

1. Fomentar y continuar el estudio sobre la teoría de agentes como una opción más en el campo de la automatización, sobre todo en el control de procesos.
2. Gestionar los medios necesarios, como la adquisición de la herramienta de desarrollo para la ejecución del proyecto relacionado con el SMA propuesto.
3. Iniciar proyectos relacionados con SMA experimentando con la metodología PROHA que es otra fuerte candidata para las aplicaciones industriales.
4. Incluir el presente trabajo como base material de estudio para los estudiantes de Ingeniería en Control Automático en la Universidad Central.

REFERENCIAS BIBLIOGRÁFICAS

BARAHONA (2011). Identificación de Sistemas Usando Matlab. Escuela superior politécnica del litoral, Laboratorio de control automático.

Barber, F., V. J. Botti, et al. (2002). Presentación IA: pasado, presente y futuro. NOVATICA. Zürich. 159.

Bellifemine, F., A. Poggi, et al. (2001). JADE: a FIPA2000 compliant agent development environment. Proceedings of the fifth international conference on Autonomous agents, ACM.

Blanco, L. J. (2002). Notas para un curso de inteligencia Artificial y Sistemas Expertos. PREGUER, La Habana.

Botero, H. A. and J. M. Ramírez (2008). Identificación de sistemas de excitación: Análisis detallado de metodología y resultados Dyna. Medellín, Año 75: 65-77.

Botti, V., V. Julián, et al. (1999). Modelling agents in hard real-time environments. MAAMAW, Springer-Verlag.

Bratman, M. E. (1987). Intentions, Plans, and Practical Reason. Libro Completo, Harvard University Press.

Bussmann, S., N. R. Jennings, et al. (2004). Multiagent Systems for Manufacturing Control. A Design Methodology. Springer Series on Agent Technology, Berlín, Springer-Verlag.

Caire, G., F. Leal, et al. (2002). "Agent Oriented Analysis using MESSAGE/UML." Springer Verlag LNCS 2222.

Cenjor, A. and A. García (2005). Control Basado en Agentes Mejorados con la Tecnología Auto-ID. RIAI. 2.

Choque , G. (2009). "Agentes Inteligentes: El siguiente pasó en la inteligencia artificial." Universidad Gerardo Valdez Balcazar

Collis, J. C. and D. T. Ndumu (1999). "The Role Modelling Guide." Applied Research and Technology BT Labs.

Compuequipos (2009). Descripción Matemática de un Proceso Dinámico Mediante la Identificación a Partir de Datos Experimentales. Asesoría en Medición y Control. Colombia.

De Giacomo, G., Y. Lespérance, et al. (2000). ConGolog, a concurrent programming language based on the situation calculus. Artificial Intelligence. 121: 109-169.

DeLoach, S. A. (2001). Analysis and Design using MaSE and agentTool. Midwest Artificial Intelligence and Cognitive Science Conference, Miami University.

Demazeau, Y. (1995). From cognitive interactions to collective behaviour in agent-based systems. European Conference on Cognitive Science, Actas de conferencia.

Ercoli, E. C. (1998). Tratamientos Biológicos. Curso latinoamericano de especialización en técnicas de remediación ambiental, AIDIS.

EURESCOM (2000). MESSAGE: Methodology for engineering systems of software agents. Initial methodology. EURESCOM. T. R. P907-D1.

EURESCOM (2001). MESSAGE: Methodology for engineering systems of software agents. EURESCOM. T. R. P907-T11.

García , A. and S. Ossowski (2003). Inteligencia Artificial Distribuida y Sistemas Multiagente. Departamentos de Inteligencia Articial y de Informática, Universidad Politécnica y Universidad Juan Carlos de Madrid.

Genesereth, M. R. (1992). An agent-based approach to software interoperability. In Proceedings of the DARPA Software Technology Conference.

Gilbert, A. (1995). The Role of Intelligent agents in the Information Infrastructure. IBM. United States.

Gómez , J. and R. Fuentes (2002). The INGENIAS Methodology. Fourth Iberoamerican Workshop on Multi-Agent Systems Iberagents

Gómez , J. and J. Pavón (2004). Elementos de INGENIAS. Desarrollo de Sistemas Multi-Agente con INGENIAS. Universidad Complutense Madrid, Dep. de Sistemas Informáticos y Programación.

Gómez , J. J. (2002). Modelado de Sistemas Multi-Agente. Tesis doctoral, Universidad Complutense de Madrid.

Gómez , J. J. (2003). Metodologías para el desarrollo de sistemas multi-agente. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. Madrid, AEPIA: 51-63.

González , E. J. (2004). Diseño e implementación de una arquitectura multipropósito basada en agentes inteligentes: aplicación a la planificación automática de agendas y al control de procesos. Tesis doctoral, Universidad de la Laguna.

González, E. J., A. Hamilton, et al. (2006). Diseño e Implementación de un Sistema Multiagente para la Identificación y Control de Procesos. Revista Ingeniería Informática. Tenerife, Spain, Universidad de La Laguna.

Guzmán, Karin, et al. (1998). Co-Tratamiento Aerobio de aguas residuales industriales y domesticas. Asociación Peruana de Ingenieria Sanitaria y Ambiental: 1-16.

Hípola, P., B. Vargas, et al. (1999) "Agentes inteligentes: definicion y tipologia. Los agentes de informacion." El profesional de la información.

Iglesias , C., M. Mercedes , et al. (1998). Analysis and design of multiagent systems using MASCommonKADS, en Intelligent Agents IV. LNAI. Berlin, SpringerVerlag. 1365 ed.

Iglesias, C. Á. (1998). Definicion de una metodología para el desarrollo de Sistemas Multi-Agente. Tesis doctoral, Universidad Politécnica de Madrid.

Inglada, J. (2002). RT-MESSAGE: Desarrollo de Sistemas Multiagente de Tiempo Real. PhD thesis, Universidad Politécnica de Valencia.

Inglada, J. and V. Botti (2002). Developing real-time multi-agent systems. Iberoamerican Worrkshop on Multi-Agent Systems Málaga.

Jacobson, I., J. Rumbaugh, et al. (1999). The Unified Software Development Process Addison-Wesley.

Julian, V. and V. Botti (2004). Developing real-time multi-agent systems. Integrated Computer-Aided Engineering, IOS Press.

Julián, V. and V. Botti (2000). Agentes Inteligentes:el siguiente paso en la Inteligencia Artificial. NOVATICA, ATI. Especial 25 aniversario: 95-99.

Julián, V., C. Carrascosa, et al. (2002). Simba: an Approach for real Time Multi-Agent Systems. V Conferencia Catalana d´Intel.ligència Artificial., Castelló, Springer-Verlag.

Julián, V. J. and V. J. Botti (2003). Estudio de Métodos de Desarrollo de sistemas multiagente. Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial. España, Asociación Española para Inteligencia Artificial. 7: 65-80.

Kinny, D., M. Georgeff, et al. (1997). A Methodology and Modelling Technique for Systems of BDI Agents. Informe.

López , M. E. (2009). Identificación de Sistemas. Aplicación al modelado de un motor de continua Universidad de Alcalá de Henares.

Lyytinen, K. S. and M. Rossi (1999). "METAEDIT+ A fully configurable Multi-User and Multi-tool CASE and CAME Environment." Springer Verlag. LGNS 1080.

MAGMA. (2003). "MAGMA Research Group." from http://www-leibniz.imag.fr/MAGMA/.

Mancilla , L. E. (2008). ¿Qué son los agentes inteligentes de software? Gaceta Ide@s CONCYTEG 25-44.

Manrique, V. (2003). Tratamiento de aguas residuales industriales. Asesoría consultoría, ANALIZA CALIDAD ®.

Mendiburu , H. A. (2003). Automatización Medioambiental. Lima, Perú-MMIII, INDECOPI.

Muñoz , C. (2009). Diseño e implementación de un servidor de vídeo adaptativo en simulador de redes ad-hoc, Universitat Politècnica de Catalunya.

Nwana, H. S., D. T. Ndumu, et al. (1999). "ZEUS: A Toolkit for Building Distributed Multi-Agent Systems." Applied Artificial Intelligence 1.

Pople, H. E. (1985). Artificial Intelligence: Problems and Issues. Joint Conference CAIA.

Pressman, R. S. (1982). Software Engineering: A Practitioner's Approach, McGraw-Hill,Inc.

Ricordel, P.-M. (2001). Programmation Orientée Multi-Agents , Développement et Déploiement de Systèmes Multi-Agents Voyelles. Tesis doctoral, Institut National Polytechnique de Grenoble.

Rodríguez , C. (2009). Sistema de Vigilancia Tecnológica y Agentes Inteligentes. Tesis doctoral, Universidad Complutense de Madrid.

Romero , M. d. C. (2010). Programa de doctorado Informática Industrial. Sistemas MultiAgente (MAS). Universidad de Sevilla, Departamento de Tecnología Electrónica 57.

Russell, S. and P. Norvig (1996). Inteligencia Artificial, un enfoque moderno. México, Prentice Hall

Shaw, M., R. DeLine, et al. (1995). "Abstractions for Software Architecture and Tools to Support Them." Carnegie Mellon University. Pittsburgh PA.

Soler, J., V. Julián, et al. (2002). Towards a real-time MAS architecture. Challenges in Open Agent Systems, Bolonia, Italia.

Tansley, D. S. W. and C. C. Hayball (1993). Knowledge Based systems Analysis and Design a KADS developer's handbook., Prentice Hall.

Union, I. T. (2002). "Formal description techniques (FDT) – Message Sequence Chart (MSC)." Telecomunication Standardization Sector of ITU.

Union, I. T. (2002). "Formal description techniques (FDT) – Specification and Description Language (SDL)." Telecomunication Standardization Sector of ITU.

Van Brussel, H., J. Wyns, et al. (1998). Reference Architecture for Holonic Manufacturing Systems: PROSA. Computer in Industry. 37: 255-274.

Wooldridge , M. and N. Jennings (1995). Intelligent Agents: Theory and Practice. . Knowledge Engineering Review. 10: 115-152.

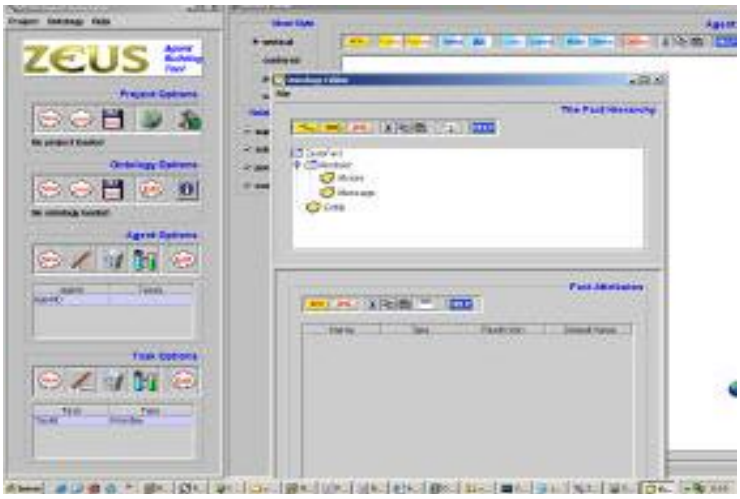
Wooldridge, M., N. R. Jennings, et al. (1998). "A Methodoly for Agent-Oriented Analysis and Design." Seattle WA.

Wooldridge, M., N. R. Jennings, et al. (2000). "The Gaia Methodology for Agent-Oriented Analysis and Design." Journal of Autonomous Agents and Multi-Agent Systems 15 2000.

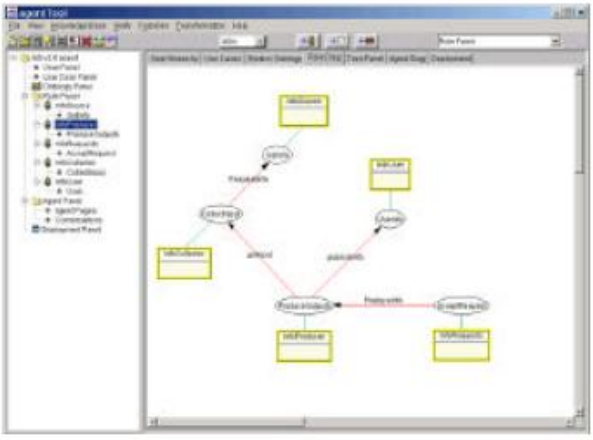
Zambonelly, F., W. M, et al. (2000). "Organisational Rules as an Abstraction for the Analysis and Design of Multi-Agent Systems." International Journal of Software Engineering and knowledge Engineering.

ANEXOS

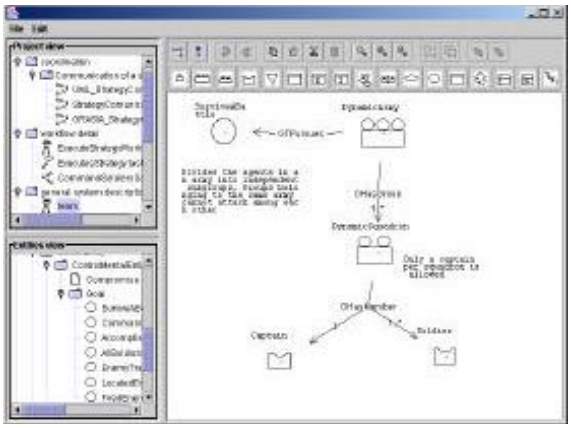
Anexo 1a. Entorno de desarrollo ZEUS.



Anexo 1b AgentTool, la herramienta de soporte de MaSE

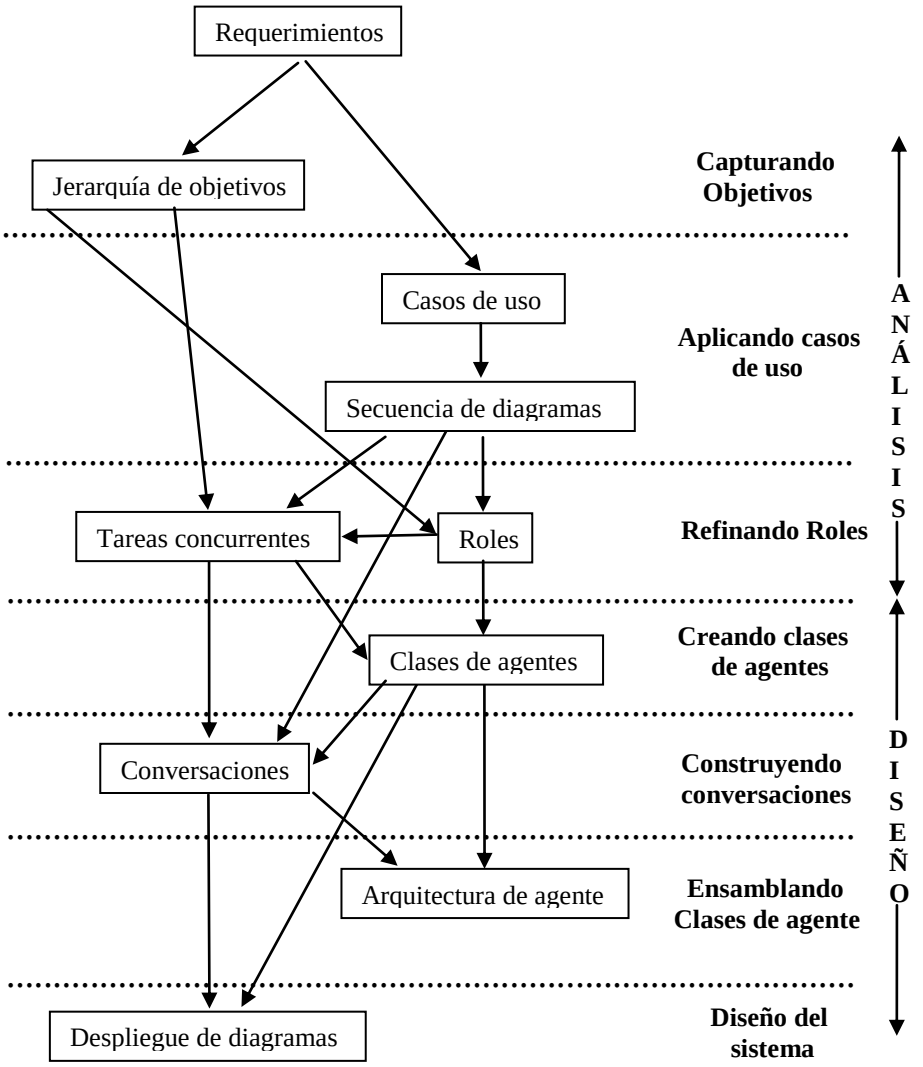


Anexo 1c. Herramienta de soporte para INGENIAS

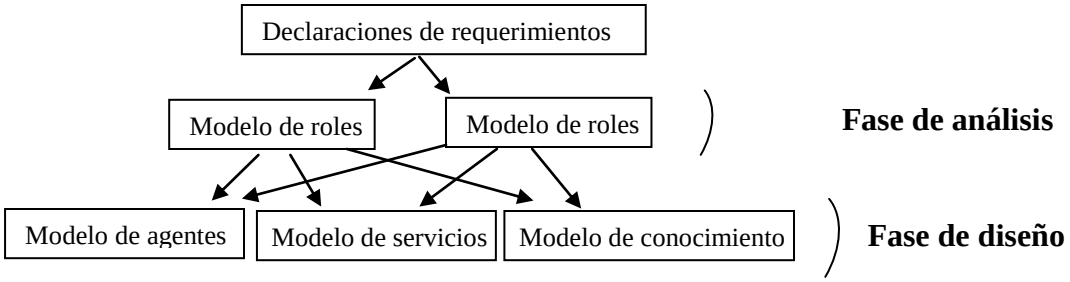


ANEXO 2

Anexo 2a. Proceso de desarrollo de la metodología MaSE.



Anexo 2b. Concepto básico de la metodología GAIA.



Anexo 3. Comparación entre las metodologías de acuerdo a orientación, herramienta de desarrollo y principales ventajas.

Metodología	Orientación	Herramienta de desarrollo	Puntos débiles	Principales ventajas
vowel engineering	Agentes	No	Proceso de desarrollo	-Modela sistemas a partir de la combinación de diferentes aspectos - Utilización de lenguajes de arquitecturas.
MAS-CommonKADS	Sistemas basados en conocimiento	Sí	Al tener la especificación en lenguaje natural, se dificulta el análisis automático de la especificación generada.	-Exhaustiva a la hora de detallar el sistema y consecuente con que el proceso de desarrollo
BDI	Agentes	No	La forma en que tienen lugar las realimentaciones no se llega a mostrar con detalle.	-Sus modelos formales.
ZEUS	Agentes	Sí	Metodológicamente es pobre.	-Referencia de cómo debe ser una herramienta para el desarrollo de SMA
MaSE	Objetos	Sí	-Su forma de modelar obvia que, como en el modelo BDI, se tienen dependencias entre los diagramas propuestos. -Cambios en el código de componentes generados implican la recopilación necesaria de la herramienta completa.	-Se aprende rápido: sólo hay que experimentar con la herramienta. -Soporte de herramientas.

GAIA	Agentes	No	-Queda a un nivel de abstracción demasiado alto. - Omite las distintas dependencias entre los modelos propuestos	-"Consigue mayor precisión"
INGENIAS	Agentes	Sí	-Se proyecta en Meta-modelos, ello atenta en su contra a la hora de aplicaciones de desarrollos reducidos.	-La integración en las prácticas de ingeniería. -Proceso de generación de código es más flexible
RT-MESSAGE	Agentes	Sí	-Las actividades definidas no se adecúan a las necesidades reales y no se indicó cómo encajaban dentro de este proceso. -Faltó trabajo en el estudio de las interdependencias entre los distintos modelos propuestos.	-Construcción de Sistemas Multiagente de Tiempo Real.
PROHA	Agentes	Sí	-Alto grado de abstracción. -Requiere observar el sistema productivo de forma global y en pleno funcionamiento,	- Se adapta a los sistemas de grandes dimensiones y alta complejidad